TOMBPC.DAT FILE FORMAT

FOR TOMB RAIDER II GAMES

Document Version 1.0

By IceBerg, Lisbon, Portugal, European Union

September 30, 2005

Disclaimer

This document is targeted to researchers and reflects the author's opinion about the internal structure of the script files. There may be errors or misinterpretations in this research, so please use with caution and please do report your own findings at the EZBoard Forum where most Tomb Raider researchers can be found (http://pub19.ezboard.com/ftreditingzonefrm2). Please do not hesitate in making an addenda or errata to this document, or in publishing your own document.

Blank page

Foreword to version 1.0

Creating scripts for Custom Levels for any of the TR games is an important part of authoring. Unfortunately information about TR scripts is scarce and sometimes misleading. Aside the TRosettaStone I couldn't find any file formats and, even there, only about TR2 scripts. Except for the TR4 area which is well covered by the data published by Core Design for TRLE. So I decided to start an independent research on the script formats for all the TR games, starting with TR2.

Most of the work contained herein was based on the TRosettaStone document and on studying Core Design's Gameflow compiler. Custom rooms were built for testing purposes.

Thanks to all those authors and researchers who have been contributing with bits and pieces of information, here and there, about the TR file formats. I've been taking notes ☺. Please do carry on. By publishing this document about scripting, I'm trying to give back to the community some of the precious help I've been receiving from it. May this effort be useful to other researchers.

IceBerg
Lisbon, Portugal
2005-09-30

Introduction

A new nomenclature, following recent orientations in the computer industry, is used to describe some integer and real variable types[1]:

sint8	: **S**igned **8**-bits	[-128..127]
sint16	: **S**igned **16**-bits	[-32768..32767]
sint32	: **S**igned **32**-bits	[-2147483648..2147483647]
uint8	: **U**nsigned **8**-bits	[0..255]
uint16	: **U**nsigned **16**-bits	[0..65535]
uint32	: **U**nsigned **32**-bits	[0..4294967295]
single	: single-precision floating point **32**-bits	
double	: double-precision floating point **64**-bits	

In the text, used occasionally, references may be made to the traditional **bytes**, **words** and **dwords**, respectively **8**, **16** and **32** bit unsigned integers, mostly to define sizes.

Software:
Core Design's GameFlow compiler; Turbo Pascal's and Vinc@eborg's TombPCEditor v1.2; Turbo Pascal's DXTRE3D v2.0 Rev2C Room Level Editor; IceBerg's HexDump and an ordinary text editor; were used together as research tools to help me understanding the TR1/2/3/4/5 script file formats.

Bibliography:
Data published with the German version of the TR2 Gold demo in 1999; the TRosettaStone dated November 1999.

Web sites:
Stella's Tomb Raider site, www.tombraider.net, was a source for some general info about the Tomb Raider universe.

The making of this document:
Page layout and all the document processing was done with Open Office, an OSS - Open Source Software distributed free and endorsed by Sun Microsystems. A fantastic publishing suite, thanks to which I'm definitely drifting away from Microsoft's Office products.

[1] *This is the same nomenclature used in the TR WAD FILE FORMAT document, series 2.*

Blank page

TR II SCRIPTING

Blank page

The Tomb Raider II TOMBPC.DAT file format

The TR2 Series has a separate script file, TOMBPC.DAT, an encrypted file containing the flow of the game and the game' strings. It controls which levels are played and in which order, what items Lara is given at the beginning of the level, what items are given upon finding all the secrets, it stores the file names of the levels and their titles, it stores the text for the options in the menus, etc. The TR2 Series also has a separate file for sounds, MAIN.SFX, except in some demos like the “The Great Wall” and the “Venice” distributions.
For example, the “Premier Collection” boxed edition of “Tomb Raider II – Golden Mask” contains:

TOMB2.EXE	912.896 bytes	11/DEC/1997
MAIN.SFX	7.961.208 bytes	05/NOV/1997
TOMBPC.DAT	5.340 bytes	05/NOV/1997
T2GOLD.EXE	920.064 bytes	19/MAY/1999
MAIN.SFX	7.609.670 bytes	17/MAY/1999
TOMBPC.DAT	2.712 bytes	19/MAY/1999

Originated from two text files, one with the script options and another with the language strings, the final DAT file was compiled by Core Design with an utility called GAMEFLOW.EXE, which was distributed by Eidos in the German demo edition of Tomb Raider II Gold. This utility will combine the two text files into one binary file and will encrypt the strings.

GAMEFLOW.EXE	85.504 bytes	31/MAR/1999

Some demo distributions of TR2 use a DEMOPC.DAT[2] file which uses the same file format as the standard DAT file. Other use a standard TOMBPC.DAT[3] file.

“The Great Wall”		
TOMB2.EXE	869.888 bytes	30/OCT/1997
DEMOPC.DAT	2.052 bytes	31/OCT/1997
“Venice”		
TOMB2.EXE	909.824 bytes	16/APR/1998
DEMOPC.DAT	2.064 bytes	16/APR/1998
“Der kalte, kalte Krieg” (German version of “The Cold War”)		
TOMB2.EXE	919.552 bytes	21/APR/1999
MAIN.SFX	7.961.208 bytes	05/NOV/1997
TOMBPC.DAT	2.907 bytes	22/APR/1999
“The Cold War”		
TOMB2.EXE	948.736 bytes	06/APR/1999
MAIN.SFX	7.961.208 bytes	05/NOV/1997
TOMBPC.DAT	2.203 bytes	31/MAR/1999
“Fool's Gold”		
T2GOLD.EXE	916.992 bytes	27/MAY/1999
MAIN.SFX	7.609.670 bytes	19/MAY/1999
TOMBPC.DAT	2.219 bytes	27/MAY/1999

[2] *These do not have a MAIN.SFX sound file, the sounds are stored in the demo level. For example, the game's BOAT.TR2 has 3,76 MB and the demo's BOAT.TR2 has 5,64 MB.*
[3] *These use the same MAIN.SFX sound files as the game distributions, matching the name of the Engine.*

The **Tomb Raider II – DAGGER OF XIAN** game contains 17 levels[4], whose titles and files are:

CHINA:		
	The Great Wall	WALL.TR2
ITALY:		
	Venice	BOAT.TR2
	Bartoli's Hideout	VENICE.TR2
	Opera House	OPERA.TR2
OCEAN:		
	Offshore Rig	RIG.TR2
	Diving Area	PLATFORM.TR2
	40 Fathoms	UNWATER.TR2
	Wreck of the Maria Doria	KEEL.TR2
	Living Quarters	LIVING.TR2
	The Deck	DECK.TR2
TIBET:		
	Tibetan Foothills	SKIDOO.TR2
	Barkhang Monastery	MONASTRY.TR2
	Catacombs of the Talion	CATACOMB.TR2
	Ice Palace	ICECAVE.TR2
CHINA:		
	Temple of Xian	EMPRTOMB.TR2
	Floating Islands	FLOATING.TR2
	The Dragon's Lair	XIAN.TR2

The **Tomb Raider II – THE GOLDEN MASK**[5] game contains 4 extra levels[6]:

BERING SEA:		
	The Cold War	LEVEL1.TR2
	Fool's Gold	LEVEL2.TR2
	Furnace of the Gods	LEVEL3.TR2
	Kingdom	LEVEL4.TR2

How all this is controlled and a lot more is stored in the TOMBPC.DAT compiled script file.
The study in depth of TR2 scripts was made possible by the files published with Core's compiler, GAMEFLOW.EXE, in particular the PCFinal.TXT and the Strings.TXT files.

[4] *Plus the separate training level located in England, Lara's Home, ASSAULT.TR2, and the Bonus Level, Home Sweet Home, HOUSE.TR2, in that same location. There is another file, TITLE.TR2, which is not a playable level, it just carries data for the game's user interface.*

[5] *Also referred to, in some literature, as Tomb Raider Gold – The Golden Mask of Tornarsuk.*

[6] *Plus a playable Bonus Level located in the United States, Nightmare In Vegas, LEVEL5.TR2, plus the non-playable title level TITLE.TR2 which carries data for the game's user interface.*

Three areas can be defined in the TOMBPC.DAT file: the header, the game flow, the strings.

The **Header** has a fixed size of 262 bytes and contains the script's version, a description of the game which usually is Core Design' signature, and the size of the game flow data that follows.

The **Gameflow** data has a fixed size of 128 bytes and controls the behaviour of the game and stores the number of elements it contains.

The **Strings** package stores the titles of the levels and the names of the files used by the game, among other game and PC strings. It also stores the actual sequence of the levels and the script commands.

The strings stored in this package may be XOR encrypted with the **Cypher_Code** byte, or not.
To find out, the **Use_Security_Tag** bit in the **Flags** field needs to be inspected. If this bit is set then the strings are encrypted, otherwise they are stored as plain text.

If the strings are stored as plain text, then they are a list of zero-terminated strings.
If the strings are XOR encrypted, then they are ... “cypher_code” terminated strings! This is because the zero is also being encrypted, and xoring the cypher with zero yields the cypher. To recover the plain text strings, *the whole list* needs to be xored.

Before each string list there is a numerical list storing the offsets of each string, and a numerical field storing the size of the list. These numerical fields are not encrypted.

Some of these strings are file names. Usually, these file names are relative to the position of the Engine, but they can be absolute path names, provided that the file remains in the same disk as the Engine. Access to a different disk is not allowed.
Pictures are usually stored in the “data” folder and have a PCX file format.
Full Motion Videos are usually stored in the “fmv” folder and have a RPL file format.
The levels and cut scene levels are usually stored in the “data” folder and have a TR2 file format.

The script commands for each and all the levels are stored in one single binary package.
Before this binary package there is a numerical list storing the offsets of each command, and a numerical field storing the size of the package. These fields are not encrypted. Each command has an OpCode which may have one Operand, or none.

The DAT file finishes with the game strings, followed by the PC strings, followed by the puzzle, pick up and key strings. All these may be encrypted or not.

The strings package has a variable size which can be deduced from the previous sizes and the total size of the DAT file:

Total_DAT_Size	=	Header_Size + Gameflow_Size + Strings_Data_Size
Header_Size	=	4 + 256 + 2 = 262
Gameflow_Size	=	128
Strings_Data_Size	=	Total_DAT_Size – 262 – 128

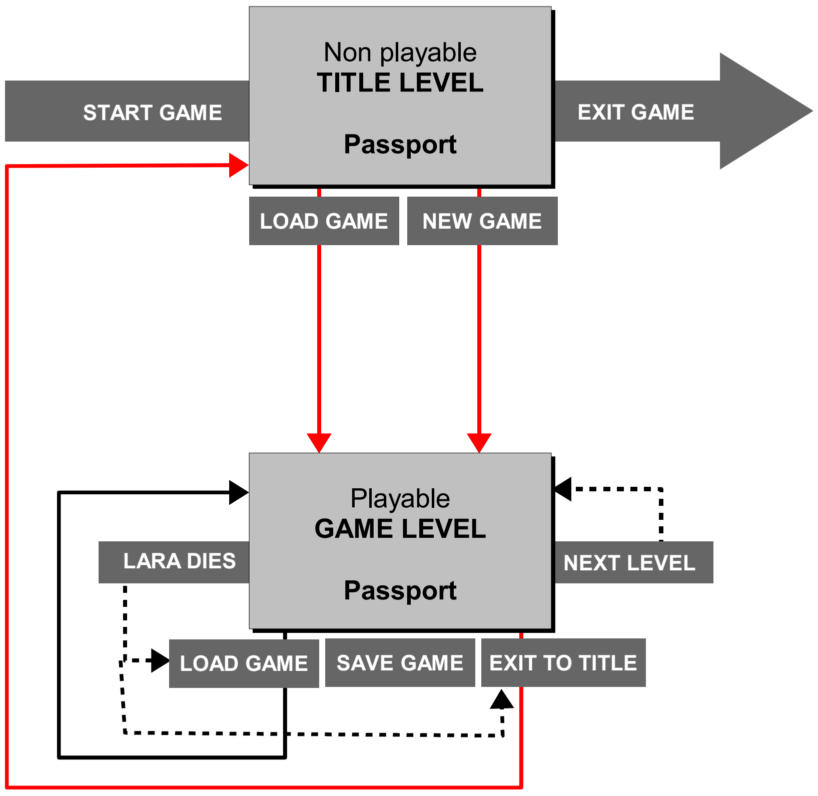
Header	**Gameflow**	**Strings**
262 bytes	128 bytes	Total - 390 bytes

The flow of the game is determined by the Engine reacting to events in the game, or by the User's choices through the game's User Interface. Such interface is provided by a special level, the TITLE.TR2 level, or by the playable levels. All these levels store the meshes, textures and animations needed to create the Title Ring, or the Option Ring, Inventory Ring and Items Ring.

A standard game will start with the **Title Ring** displayed on top of the TITLE.PCX picture.
The Passport, Controls, Sound and possibly the Polaroid, are options available in this Ring.
These elements are stored in the TITLE.TR2 non-playable level.

As the game progresses, the User Interface will also show the elements contained in the playable levels. The **Option Ring** contains the Passport, Controls and Sound options. The **Inventory Ring** contains the Chronometer for the statistics, the Weapons and the Supplies (Medi Packs and Flares). The **Items Ring** contains the objects that Lara collects to solve puzzles.

From the Title Ring either Load Game or New Game will result in a level being played.
Level playing can end for different reasons. The User presses the ESC key, or Lara dies, or Lara reaches the end of the level. If the level is finished, a panel with some statistics will show up, past which the next level in the script is loaded and played. If Lara dies, the game exits to the Option Ring with a GAME OVER title and the choices are Load Game or Exit to Title, with Save Game being not-selectable. If the User called the Inventory Ring he may be choosing a weapon, he may go to the Items Ring or he may go to the Option Ring. Reaching the Option Ring this way will make available the options Load Game, Save Game and Exit to Title.



Section 1 – Header

Script_Version (**uint32**).
A valid Tomb Raider II DAT file has a value of **3** in this field.

Description (**256 bytes**).
This is a fixed length field, usually containing a null-terminated string describing the game and showing Core Design' signature. The unused part of this field contains zeros[7].

These are the signatures found in the distributed games and demos[8]:

Game or Demo	Description
Tomb Raider II – Dagger of Xian	Tomb Raider II Script. Final Release Version 1.1 (c) Core Design Ltd 1997
Tomb Raider II – Golden Mask	Tomb Raider II Script. Final Release Version 1.1 (c) Core Design Ltd 1997
Tomb Raider II – Demo 1 The Great Wall	Tomb Raider II Script. Mag Preview (c) Core Design Ltd 1997
Tomb Raider II – Demo 2 Venice	Tomb Raider II Script. Internet Demo 15/4/98 (c) Core Design Ltd 1998
Tomb Raider II GOLD – Demo 1 The Cold War	Tomb Raider II Script. Final Release Version 1.1 (c) Core Design Ltd 1997
Tomb Raider II GOLD – Demo 2 Fool's Gold	Tomb Raider II Script. Final Release Version 1.1 (c) Core Design Ltd 1997

Gameflow_Size (**uint16**)[9].
This field contains the size, in bytes, of the game flow data that follows, always **128** bytes. It may also be seen as an offset to the strings data located after the game flow data.

[7] *There can be more then one zero-terminated string stored here, in case there is an application capable of reading them. Otherwise the whole field may be ignored.*
[8] *The default signature stored in Core Design's GAMEFLOW.EXE compiler is:*
Tomb Raider II PC Internal Development Version (c) Core Design Ltd 1997
[9] *This field is missing in the TRosettaStone document.*

Blank page

Section 2 – Gameflow

FirstOption (**uint32**).
What to do when the game starts, past the LEGAL.PCX and the LOGO.RPL full motion video. When the game flow reaches that point, it can go to the Title Screen and its Title Ring, or it can go directly to one of the Levels available, without even displaying the Title Ring.
The default is EXIT_TO_TITLE ($0000 0500).

Title_Replace (**sint32**).
Defines what happens when an EXIT_TO_TITLE is requested. The Title Screen may be available or not. That depends on the status of the **Title_Disabled** bit in the **Flags** field.
If this bit is *not* set, then the Title Screen *is available* and will display its Title Ring. Otherwise the Title Screen is not available, nor is the Title Ring. In that case, what happens upon an EXIT_TO_TITLE request? The game flow will follow the Direction stored in this field.
The default is **-1** ($FFFF FFFF), which would cause an EXITGAME or an error, but is not executed in a default situation where **Title_Disabled** is not set.

OnDeath_Demo_Mode (**uint32**).
What to do when Lara dies during the demo mode.
The default is EXIT_TO_TITLE ($0000 0500).

OnDeath_InGame (**uint32**).
What to do to when Lara dies during the game.
The default is LEVEL ($0000 0000).

NoInput_Time (**uint32**).
Time to wait before starting the demo mode. This is the number of 1/30th of a second.
If this field is enabled or not, that depends on the **NoInput_Timeout** bit in the **Flags** field.
If that bit is *not* set the game flow will start the demo after the waiting time.
If the bit *is* set the game waits forever and the demo is ignored.
The default is **900**, meaning 30 seconds.

On_Demo_Interrupt (**uint32**).
What to do when the demo mode is interrupted.
The default is EXIT_TO_TITLE ($0000 0500).

On_Demo_End (**uint32**).
What to do when the demo mode ends.
The default is EXIT_TO_TITLE ($0000 0500).

Filler_1 (**36 bytes**).
Filler bytes. This completes a sub-total of 64 bytes including this field.

Num_Levels (**uint16**).
Number of levels in the game, including the training level, not including the title level.

Num_Pictures (**uint16**).
Number of chapter screens in the game.
Not used by Tomb Raider II, but this field exists in the DAT file, anyway.

Num_Titles (**uint16**).
Number of title elements available. This includes the TITLE.TR2 level plus the legal and title pictures in *.PCX format.

Num_FMVs (**uint16**).
Number of Full Motion Videos in the game.

Num_Cutscenes (**uint16**).
Number of cut scene sequences in the game.

Num_Demos (**uint16**).
Number of demo levels in the game. These can be the same levels used by the game, or not. What matters is that these levels must contain demo sequences.

Title_Track (**uint16**).
ID of the title' soundtrack.

SingleLevel (**uint16**).
The game plays only one single level, overriding the script.
Which level? That depends on the value stored in this field. A value of **1** indicates the first level, a value of **2** indicates the second level, etc.
A value of **-1** switches OFF this command and the system remains in multi-level mode, in which case the script is executed. A value of **0** will send the game flow to the Title Screen, creating a loop with no exit – do not place a zero in this field!
The default is **-1**, meaning that the game is multi-level.

Filler_2 (**32 bytes**).
Filler bytes. This completes a sub-total of 48 bytes including this field, after the previous 64 bytes.

Flags (**uint16**).
Various flags enabling or disabling several options[10]. Treated as a bit field, it stores the following:

Enable_Cheat_Code	**Flags and $0800** (**1 bit**) Bit[**11**] apparently has no effect on the PC game. The known *flare/step/step/rotate/jump* cheat sequence does not depend on this bit. .
Select_Any_Level	**Flags and $0400** (**1 bit**) If bit[**10**] is set, it indicates that the names of the levels are all listed in the Passport. It will be the player's choice which level is played. Otherwise levels are not displayed and the order defined in the script is kept and followed in sequence.
Unknown	**Flags and $0200** (**1 bit**) Bit[**9**] apparently has no effect on the PC game. Usually set.
Use_Security_Tag	**Flags and $0100** (**1 bit**) If bit[**8**] is set, it indicates that a cypher byte was used to encrypt the strings in the script file, and is stored in the **Cypher_Code** field.
DOZY_Cheat_Enabled	**Flags and $0080** (**1 bit**) If bit[**7**] is set, it indicates that the game has the DOZY cheat enabled, but what does that mean on a TR2 game? The known *flare/step/step/rotate/jump* cheat sequence does not depend on this field. Typing "DOZY" yields nothing.
LockOut_OptionRing	**Flags and $0040** (**1 bit**) If bit[**6**] is set, it indicates that the user has no access to the Option Ring while playing the game. This means that the user has no access to the Passport, therefore cannot use save games or exit the game.
ScreenSizing_Disabled	**Flags and $0020** (**1 bit**) If bit[**5**] is set, it indicates that the game does not allow screen resizing. The F1 / F2 / F3 / F4 keys will not work, they will not change the screen resolution or the screen size. F12 still toggles between full screen and windowed.
LoadSave_Disabled	**Flags and $0010** (**1 bit**) If bit[**4**] is set, it indicates that the game does not allow save games, but the consequences of this do require some extra explanations. The Title Screen will have a Passport which only allows EXITGAME. During the game, if the user calls the Passport, it only allows EXIT_TO_TITLE. Given these restrictions, a game that starts with the Title Screen cannot be actually played – it exits. Such a game must start with a level which must be declared as the **FirstOption**. During the game the F5 / F6 keys do not save / load.

[10] *Maybe some of these fields, which do not work on the PC, are for the PSX console.*

NoInput_Timeout **Flags and $0008** (**1 bit**)
If bit[**3**] is set, it indicates that the game waits forever if there is no input. Otherwise it returns to the Title Screen, or it enters a demo mode, after a certain period of time with no user input.

CheatModeCheck_Disabled **Flags and $0004** (**1 bit**)
If bit[**2**] is set, it indicates that the game does *not* look for the cheat sequence keystrokes and events.
This will disable the *flare/step/step/rotate/jump* sequence.

Title_Disabled **Flags and $0002** (**1 bit**)
If bit[**1**] is set, it indicates that the game has no Title Screen. This affects the way how game flow operates, in conjunction with **Title_Replace**.

DemoVersion **Flags and $0001** (**1 bit**)
If bit[**0**] is set, it indicates that the game is a demo distribution. Otherwise it is a normal game distribution. What difference does it make? This flag was found only in the Wall and Venice demos, which have a DEMOPC.DAT file as a script, and do not use the MAIN.SFX sound file.

				Enable_Cheat_Code	Select_Any_Level	Unknown	Use_Security_Tag	DOZY_Cheat_Enabled	LockOut_OptionRing	ScreenSizing_Disabled	LoadSave_Disabled	NoInput_Timeout	CheatModeCheck_Disabled	Title_Disabled	DemoVersion
$8000	$4000	$2000	$1000	$0800	$0400	$0200	$0100	$0080	$0040	$0020	$0010	$0008	$0004	$0002	$0001
0	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

The settings above are the typical ones for a regular game. The **Unknown** bit is set in all the games and demos, except in the “The Great Wall” and the “Venice” demos. The cypher byte is optional, but every DAT file has been encrypted, therefore **Use_Security_Tag** is set. The value obtained this way is **$0300**, which can be considered as the default value for the **Flags** field.

Filler_3 (**6 bytes**).
Filler bytes. This completes a sub-total of 8 bytes including this field, after the previous 64 + 48 bytes.

Cypher_Code (**uint8**).
Cypher **byte** used to encrypt the strings in the script file. Core Design used the value **166** (**$A6**) to XOR the TR2 strings. If this command is added to the script, then the GAMEFLOW.EXE compiler will set the **Use_Security_Tag** bit in the **Flags** field.

Language (**uint8**).
Language **byte**. There are five valid strings that can be used in the script:

ENGLISH	0
FRENCH	1
German	2
AMERICAN	3
JAPANESE	4

If a meaningless string is placed here the GAMEFLOW.EXE utility compiles it as **1** if the first letter is **<= J**, and it compiles as **255** if the first letter is **> J**.

Secret_Track (**uint16**).
ID of the “found a secret” soundtrack.

Filler_4 (**4 bytes**).
Filler bytes. This completes a sub-total of 8 bytes including this field, after the previous sub-totals of 64 + 48 + 8 bytes, making a grand total of **128** bytes, the **Gameflow_Data_Size**.

Blank page

Section 3 – Strings

Level_Names_Offset_List (Num_Levels * 2 bytes)
Each record in this list is a **uint16** value representing an offset to the name of a level, including the training level, not including the title level.

Level_Names_Num_Bytes (uint16)
Size of the **Level_Names_List**, expressed in **bytes**.

Level_Names_List (Level_Names_Num_Bytes)
The names of the levels are stored in this list.

Picture_Filenames_Offset_List (Num_Pictures * 2 bytes)
Each record in this list is a **uint16** value representing an offset to the file name of a picture.
Not used in the TR2 Series[11].

Picture_Filenames_Num_Bytes (uint16)
Size of the **Picture_FileNames_List** expressed in **bytes**.
Not used in the TR2 Series.

Picture_Filenames_List(Picture_Filenames_Num_Bytes)
The file names of the chapter screen pictures are stored in this list.
Not used in the TR2 Series.

Title_Filenames_Offset_List (Num_Titles * 2 bytes)
Each record in this list is a **uint16** value representing an offset to the file name of a title element. The first title element in the list *must* always be TITLE.TR2, followed by the PCX pictures for the Title Screen and the Legal Screen, for the UK, USA and Japan.

Title_Filenames_Num_Bytes (uint16)
Size of the **Title_FileNames_List**, expressed in **bytes**.

Title_Filenames_List (Title_Filenames_Num_Bytes)
The file names of the title elements are stored in this list.

[11] *If the script text contains the OpCode PICTURE ($0000) in any of the level sequences, then the GAMEFLOW.EXE utility will compile it, and the Picture_Filenames fields will have some data in them. However, OpCode PICTURE yields nothing in-game. Or, if OpCode LOAD_PIC ($000C) is declared in any level sequence, the GAMEFLOW.EXE utility does not even compile it at all. In both circumstances, no pictures are actually used in-game, rendering the Picture_Filenames fields totally useless under TR2.*

FMV_Filenames_Offset_List (**Num_FMVs * 2 bytes**)
Each record in this list is a **uint16** value representing an offset to the file name of a Full Motion Video. These can be corporate logos or pre-rendered animated sequences.

FMV_Filenames_Num_Bytes (**uint16**)
Size of the **FMV_FileNames_List**, expressed in **bytes**.

FMV_Filenames_List (**FMV_Filenames_Num_Bytes**)
The file names of the Full Motion Videos are stored in this list.

Level_Filenames_Offset_List (**Num_Levels * 2 bytes**)
Each record in this list is a **uint16** value representing an offset to the file name of a level.

Level_Filenames_Num_Bytes (**uint16**)
Size of the **Level_FileNames_List**, expressed in **bytes**.

Level_Filenames_List (**Level_Filenames_Num_Bytes**)
The file names of the levels are stored in this list.

Cutscene_Filenames_Offset_List (**Num_Cutscenes * 2 bytes**)
Each record in this list is a **uint16** value representing an offset to the file name of a level containing a cut scene.

Cutscene_Filenames_Num_Bytes (**uint16**)
Size of the **Cutscene_FileNames_List**, expressed in **bytes**.

Cutscene_Filenames_List (**Cutscene_Filenames_Num_Bytes**)
The file names of the cut scenes are stored in this list.

Script_Offset_List ((**Num_Levels + 1**) * **2 bytes**)
Each record in this list is a **uint16** value representing an offset to a script command.
The (+ **1**) above is to include the Level sequences plus the FrontEnd sequence.

Script_Num_Bytes (**uint16**)
Size of the **Script_Package**, expressed in **bytes**.

Script_Package (**Script_Num_Bytes**)
The script sequences are stored in this package, translated into OpCodes and Operands.

GAMEFLOW.EXE script commands	OpCode	Operand	Description
PICTURE	$0000 (00)	Picture ID	Unused. Compiles but does not show in-game. Maybe PSX.
PSX_TRACK (?)	$0001 (01)	Track ID	Unused. Does not compile. Maybe PSX.
PSX_FMV (?)	$0002 (02)	FMV ID	Unused. Does not compile. Maybe PSX.
FMV	$0003 (03)	FMV ID	Display Full Motion Video.
GAME	$0004 (04)	Level ID	Start a playable level.
CUT	$0005 (05)	Cutscene ID	Display cut scene sequence.
COMPLETE	$0006 (06)	-	Display level-completion statistics panel.
DEMO, PCDEMO	$0007 (07)	Demo ID	Display demo sequence.
PSX_DEMO (?)	$0008 (08)	Demo ID	Unused. Does not compile. Maybe PSX.
END	$0009 (09)	-	Closes script sequences, LEVEL, DEMOLEVEL, GYM, etc...
TRACK, PCTRACK	$000A (10)	Track ID	Play Soundtrack (it precedes opcodes of associated levels).
SUNSET	$000B (11)	-	Unknown. Nothing changes in-game. Maybe this is an ancestor of the TR4 LensFlare command, not actually implemented under TR2.
LOAD_PIC	$000C (12)	Picture ID	Unused. Does not compile. Will be used under TR3.
DEADLY_WATER	$000D (13)	-	Unknown. Nothing changes in-game. Maybe this is an ancestor of the TR3 Death_By_Drowning effect, not actually implemented under TR2.
REMOVE_WEAPONS	$000E (14)	-	Lara starts the level with no weapons.
GAMECOMPLETE	$000F (15)	-	End of game, shows the final statistics and starts the credits sequence with music ID = 52.
CUTANGLE	$0010 (16)	HRotation	Matches the North-South orientation of the Room Editor and the North-South orientation of the 3D animated characters from a CAD application.
NOFLOOR	$0011 (17)	Depth	Death_By_Depth. Lara dies when her feet reach the given depth. If falling, 4 to 5 extra blocks are added to Depth.
STARTINV, BONUS	$0012 (18)	Item ID	Items given to Lara at level-start or at all-secrets-found.
STARTANIM	$0013 (19)	Anim ID	Special Lara's animation when the level starts.
SECRETS	$0014 (20)	OnOff	If zero, the level does not account for secrets.
KILLTOCOMPLETE	$0015 (21)	-	Kill all enemies to finish the level.
REMOVE_AMMO	$0016 (22)	-	Lara starts the level without ammunition or medi packs.

The STARTINV and the BONUS OpCodes both give items to Lara.
The amount of medi packs, flares or ammunition given is one unit per OpCode, only. In order to increase the quantity given to Lara, the OpCodes need to be repeated in the script.

GAMEFLOW.EXE script parameters	STARTINV Operand	BONUS Operand	Description
PISTOLS	1000	0	Standard pistols (2)
SHOTGUN	1001	1	Shotgun (1)
AUTOPISTOLS	1002	2	Automatic Pistols (2)
UZIS	1003	3	Uzis (2)
HARPOON	1004	4	Harpoon gun (1)
M16	1005	5	M16 (1)
ROCKET	1006	6	Grenade launcher (1)
PISTOLS_AMMO	1007	7	Pistol clip (no effect, infinite by default)
SHOTGUN_AMMO	1008	8	Shotgun-shell box (adds 2 shells)
AUTOPISTOLS_AMMO	1009	9	Automatic Pistols clip (adds 2 shells)
UZI_AMMO	1010	10	Uzi clip (adds 2 shells)
HARPOON_AMMO	1011	11	Harpoon bundle (adds 2 harpoons)
M16_AMMO	1012	12	M16 clip (adds 2 shells)
ROCKET_AMMO	1013	13	Grenade pack (adds 1 grenade)
FLARES	1014	14	Flare box (adds 1 flare)
MEDI	1015	15	Small medi pack (adds 1 pack)
BIGMEDI	1016	16	Big medi pack (adds 1 pack)
PICKUP1	1017	17	Pickup item 1
PICKUP2	1018	18	Pickup item 2
PUZZLE1	1019	19	Puzzle item 1
PUZZLE2	1020	20	Puzzle item 2
PUZZLE3	1021	21	Puzzle item 3
PUZZLE4	1022	22	Puzzle item 4
KEY1	1023	23	Key item 1
KEY2	1024	24	Key item 2
KEY3	1025	25	Key item 3
KEY4	1026	26	Key item 4

Num_Game_Strings (**uint16**)
Number of generic game strings for a TR2 game.
Although this numeric field exists here, the GAMEFLOW.EXE compiler expects to see **89** strings.

Game_Strings_Offset_List (**Num_Game_Strings * 2 bytes**)
Each record in this list is a **uint16** value representing an offset to a string.

Game_Strings_Num_Bytes (**uint16**)
Size of the **Game_Strings_List**, expressed in **bytes**.

Game_Strings_List (**Game_Strings_Num_Bytes**)
The generic game strings are stored in this list.
Their number must be **Num_Game_Strings**. If necessary pad with a dummy name, like “spare”.

PC_Strings_Offset_List (**41 * 2 bytes**)
Each record in this list is a **uint16** value representing an offset to a string.
The number of these strings is hard-coded as **41**.

PC_Strings_Num_Bytes (**uint16**)
Size of the **PC_Strings_List**, expressed in **bytes**.

PC_Strings_List (**PC_Strings_Num_Bytes**)
The generic strings are stored in this list.
Their number must be **41**. If necessary pad with a dummy name, like “spare”.

Puzzle1_Offset_List (**Num_Levels * 2 bytes**)
Each record in this list is a **uint16** value representing an offset to a string.

Puzzle1_Num_Bytes (**uint16**)
Size of the **Puzzle1_List**, expressed in **bytes**.

Puzzle1_List (**Puzzle1_Num_Bytes**)
The strings for all Puzzle1 are stored in this list.
Each level available in the game will pick up its string for Puzzle1 from this list.
The first level takes the first string, the second level takes the second string, etc.
The number of strings must be **Num_Levels**. If necessary pad with a dummy name, like “P1”.

Puzzle2_Offset_List (**Num_Levels** * **2 bytes**)
Each record in this list is a **uint16** value representing an offset to a string.

Puzzle2_Num_Bytes (**uint16**)
Size of the **Puzzle2_List**, expressed in **bytes**.

Puzzle2_List (**Puzzle2_Num_Bytes**)
The strings for all Puzzle2 are stored in this list.
Each level available in the game will pick up its string for Puzzle2 from this list.
The first level takes the first string, the second level takes the second string, etc.
The number of strings must be **Num_Levels**. If necessary pad with a dummy name, like "P2".

Puzzle3_Offset_List (**Num_Levels** * **2 bytes**)
Each record in this list is a **uint16** value representing an offset to a string.

Puzzle3_Num_Bytes (**uint16**)
Size of the **Puzzle3_List**, expressed in **bytes**.

Puzzle3_List(**Puzzle3_Num_Bytes**)
The strings for all Puzzle3 are stored in this list.
Each level available in the game will pick up its string for Puzzle3 from this list.
The first level takes the first string, the second level takes the second string, etc.
The number of strings must be **Num_Levels**. If necessary pad with a dummy name, like "P3".

Puzzle4_Offset_List (**Num_Levels** * **2 bytes**)
Each record in this list is a **uint16** value representing an offset to a string.

Puzzle4_Num_Bytes (**uint16**)
Size of the **Puzzle4_List**, expressed in **bytes**.

Puzzle4_List (**Puzzle4_Num_Bytes**)
The strings for all Puzzle4 are stored in this list.
Each level available in the game will pick up its string for Puzzle4 from this list.
The first level takes the first string, the second level takes the second string, etc.
The number of strings must be **Num_Levels**. If necessary pad with a dummy name, like "P4".

Pickup1_Offset_List (**Num_Levels** * **2 bytes**)
Each record in this list is a **uint16** value representing an offset to a string.

Pickup1_Num_Bytes (**uint16**)
Size of the **Pickup1_List**, expressed in **bytes**.

Pickup1_List (**Pickup1_Num_Bytes**)
The strings for all Pickup1 are stored in this list.
Each level available in the game will pick up its string for Pickup1 from this list.
The first level takes the first string, the second level takes the second string, etc.
The number of strings must be **Num_Levels**. If necessary pad with a dummy name, like “P1”.

Pickup2_Offset_List (**Num_Levels** * **2 bytes**)
Each record in this list is a **uint16** value representing an offset to a string.

Pickup2_Num_Bytes (**uint16**)
Size of the **Pickup2_List**, expressed in **bytes**.

Pickup2_List (**Pickup2_Num_Bytes**)
The strings for all Pickup2 are stored in this list.
Each level available in the game will pick up its string for Pickup2 from this list.
The first level takes the first string, the second level takes the second string, etc.
The number of strings must be **Num_Levels**. If necessary pad with a dummy name, like “P2”.

Key1_Offset_List (**Num_Levels** * **2 bytes**)
Each record in this list is a **uint16** value representing an offset to a string.

Key1_Num_Bytes (**uint16**)
Size of the **Key1_List**, expressed in **bytes**.

Key1_List (**Key1_Num_Bytes**)
The strings for all Key1 are stored in this list.
Each level available in the game will pick up its string for Key1 from this list.
The first level takes the first string, the second level takes the second string, etc.
The number of strings must be **Num_Levels**. If necessary pad with a dummy name, like “K1”.

Key2_Offset_List (**Num_Levels** * **2 bytes**)
Each record in this list is a **uint16** value representing an offset to a string.

Key2_Num_Bytes (**uint16**)
Size of the **Key2_List**, expressed in **bytes**.

Key2_List(**Key2_Num_Bytes**)
The strings for all Key2 are stored in this list.
Each level available in the game will pick up its string for Key2 from this list.
The first level takes the first string, the second level takes the second string, etc.
The number of strings must be **Num_Levels**. If necessary pad with a dummy name, like “K2”.

Key3_Offset_List (**Num_Levels** * **2 bytes**)
Each record in this list is a **uint16** value representing an offset to a string.

Key3_Num_Bytes (**uint16**)
Size of the **Key3_List**, expressed in **bytes**.

Key3_List (**Key3_Num_Bytes**)
The strings for all Key3 are stored in this list.
Each level available in the game will pick up its string for Key3 from this list.
The first level takes the first string, the second level takes the second string, etc.
The number of strings must be **Num_Levels**. If necessary pad with a dummy name, like “K3”.

Key4_Offset_List (**Num_Levels** * **2 bytes**)
Each record in this list is a **uint16** value representing an offset to a string.

Key4_Num_Bytes (**uint16**)
Size of the **Key4_List**, expressed in **bytes**.

Key4_List (**Key4_Num_Bytes**)
The strings for all Key4 are stored in this list.
Each level available in the game will pick up its string for Key4 from this list.
The first level takes the first string, the second level takes the second string, etc.
The number of strings must be **Num_Levels**. If necessary pad with a dummy name, like “K4”.

EXPLORING THE DAT FILE

Blank page

GAME FLOW COMPARATIVE TABLE

NOTE: ONLY THE LOWER WORD IS SHOWN HERE, WHICH IS ENOUGH FOR RESEARCH PURPOSES.	DX	GM	GM Demo	Cold War	Fool's Gold	Great Wall	Venice
FirstOption	$0500	$0500	$0500	$0500	$0500	$0001	$0001
Title_Replace	$FFFF	$FFFF	$FFFF	$FFFF	$FFFF	$0700	$0700
OnDeath_Demo_Mode	$0500	$0500	$0500	$0500	$0500	$0001	$0500
OnDeath_InGame	$0000	$0000	$0000	$0000	$0000	$0001	$0700
NoInput_Time	900	900	900	900	900	900	900
On_Demo_Interrupt	$0500	$0500	$0500	$0500	$0500	$0500	$0500
On_Demo_End	$0500	$0500	$0500	$0500	$0500	$0500	$0500
Filler_1 (**36 bytes**)							
Num_Level_Strings	22	6	6	2	2	2	2
Num_Picture_Strings	0	0	0	0	0	0	0
Num_Title_Strings	7	7	7	7	7	0	1
Num_FMV_Strings	8	1	1	1	1	0	0
Num_Cutscene_Strings	4	0	0	0	0	0	0
Num_Demo_Strings	3	0	0	0	0	0	0
Title_Track	64	64	64	64	64	2	0
SingleLevel	-1	-1	1	-1	-1	1	1
Filler_2 (**32 bytes**)							
Enable_Cheat_Code	0	0	0	0	0	0	0
Select_Any_Level	0	0	0	0	0	0	0
Unknown	1	1	1	1	1	0	0
Use_Security_Tag	1	1	1	1	1	1	1
DOZY_Cheat_Enabled	0	0	1	1	0	0	0
LockOut_OptionRing	0	0	0	0	0	0	0
ScreenSizing_Disabled	0	0	0	0	0	0	0
LoadSave_Disabled	0	0	0	0	0	0	0
NoInput_Timeout	0	0	0	0	0	0	0
CheatModeCheck_Disabled	0	0	0	0	0	1	1
Title_Disabled	0	0	0	0	0	1	1
DemoVersion	0	0	0	0	0	1	1
Filler_3 (**6 bytes**)							
Cypher_Code	$A6	$A6	$A6	$A6	$A6	$A6	$A6
Language	$00	$00	$00	$00	$00	$00	$00
Secret_Track	47	47	47	47	47	47	47
Filler_4 (**4 bytes**)							

DIRECTIONS FOR THE GAME FLOW

Looking at the **GAME FLOW COMPARATIVE TABLE**, we can see that the **Gameflow** section of the DAT file is broken down in four sub-sections, with respectively **64**, **48**, **8** and **8 bytes**, making a total of **128 bytes** of size.

The first sub-section contains the Directions for the game flow. The meaning of those hexadecimal values was found by studying the GAMEFLOW compiler and its related input (TXT) and output (LOG) text files.

VALUE	Script.TXT	Script.LOG	in GAMEFLOW.EXE
$0000	LEVEL LEVEL 0 SEQUENCE SEQUENCE 0	STARTGAME 0	**STARTGAME**
$0100	does not compile		**STARTSAVEDGAME**
$0200	does not compile		**STARTCINE**
$0300	does not compile		**STARTFMV**
$0400	DEMO DEMO 0	STARTDEMO 0	**STARTDEMO**
$0500	EXIT_TO_TITLE	EXIT_TO_TITLE	**EXIT_TO_TITLE**
$0600	does not compile		**LEVELCOMPLETE**
$0700	EXITGAME	EXITGAME	**EXITGAME**
$0800	does not compile		**EXIT_TO_OPTION**
$0900	does not compile		**TITLE_DESELECT**
$FFFF	-1		

Some of these Directions can have a range of values, like LEVEL or DEMO, which can have 256 values, [0 .. 255].

VALUE	Script.TXT	Script.LOG
$0000	LEVEL LEVEL 0 SEQUENCE SEQUENCE 0	STARTGAME 0
$0001	LEVEL 1	STARTGAME 1
$0002	LEVEL 2	STARTGAME 2
$0003	LEVEL 3	STARTGAME 3
... **$00FF**	... LEVEL 255	... STARTGAME 255

Some interpretations in the **DIRECTIONS** table were extrapolated from the order of the strings in the compiler executable file, those marked as “does not compile”. The GAMEFLOW compiler does not accept them in the input Script.TXT file. However, there is a workaround.

The input LEVEL 255 is compiled as **$00FF** and produces the output STARTGAME 255.
The workaround is that the input LEVEL 256 is compiled as **$0100**, which is the value for **STARTSAVEDGAME**. The output in the Script.LOG file is written as STARTSAVEDGAME 0, suggesting that a range is allowed.

VALUE	Script.TXT	Script.LOG
$0100	LEVEL 256	STARTSAVEDGAME 0
$0101	LEVEL 257	STARTSAVEDGAME 1
$0102	LEVEL 258	STARTSAVEDGAME 2
$0103	LEVEL 259	STARTSAVEDGAME 3
... **$01FF**	... LEVEL 511	... STARTSAVEDGAME 255

The same workaround works for other Directions as well.

The input Direction LEVEL 512 is compiled as **$0200**, which is the value for **STARTCINE**. The output in the Script.LOG file is written as STARTCINE 0, suggesting that a range is allowed.
The input Direction LEVEL 768 is compiled as **$0300**, which is the value for **STARTFMV**. The output in the Script.LOG file is written as STARTFMV 0, suggesting that a range is allowed.

No workaround is needed for the next input, DEMO, which compiles normally and also has a range of 256 values. The input DEMO 255 is compiled as **$04FF** and correctly produces the output DEMO 255.

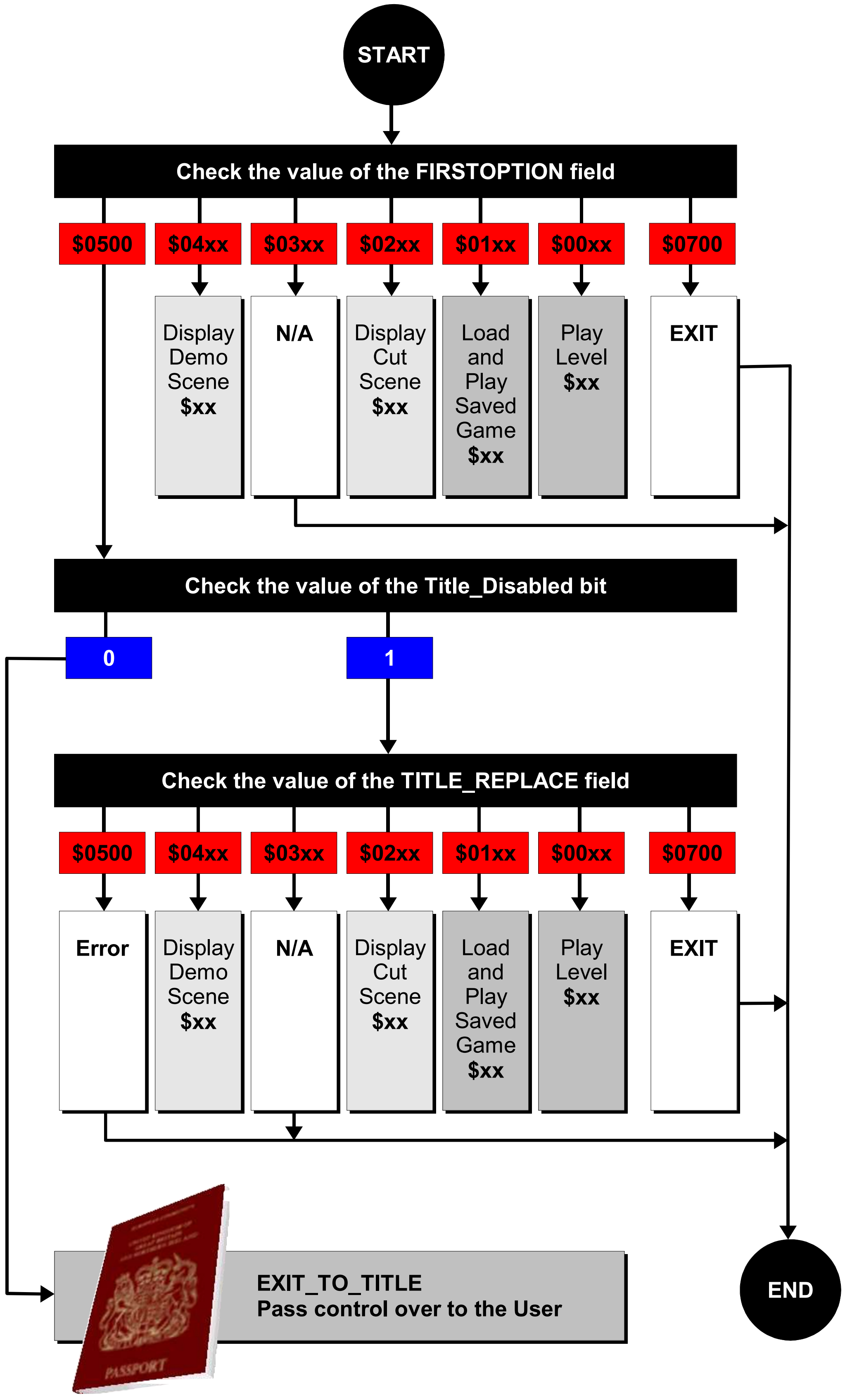
The next input, EXIT_TO_TITLE, compiles as **$0500** and produces EXIT_TO_TITLE 0 as output in the Script.LOG file. However, no ranges are allowed here. If a parameter is declared, it is ignored and the compiled value always is **$0500**.
It may be interesting to note that the workaround technique can also be used here.
The input Direction DEMO 256 is compiled the same as EXIT_TO_TITLE.
Following the same reasoning as before, the input Direction DEMO 257 is compiled as **$0501** and produces the output EXIT_TO_TITLE 1 in the Script.LOG file.

And the same reasoning carries on the same way with the other Directions. When the compiler does not accept a given Direction, it can be produced with a workaround. If the final result is interesting or not, that remains to be tested in a real situation.

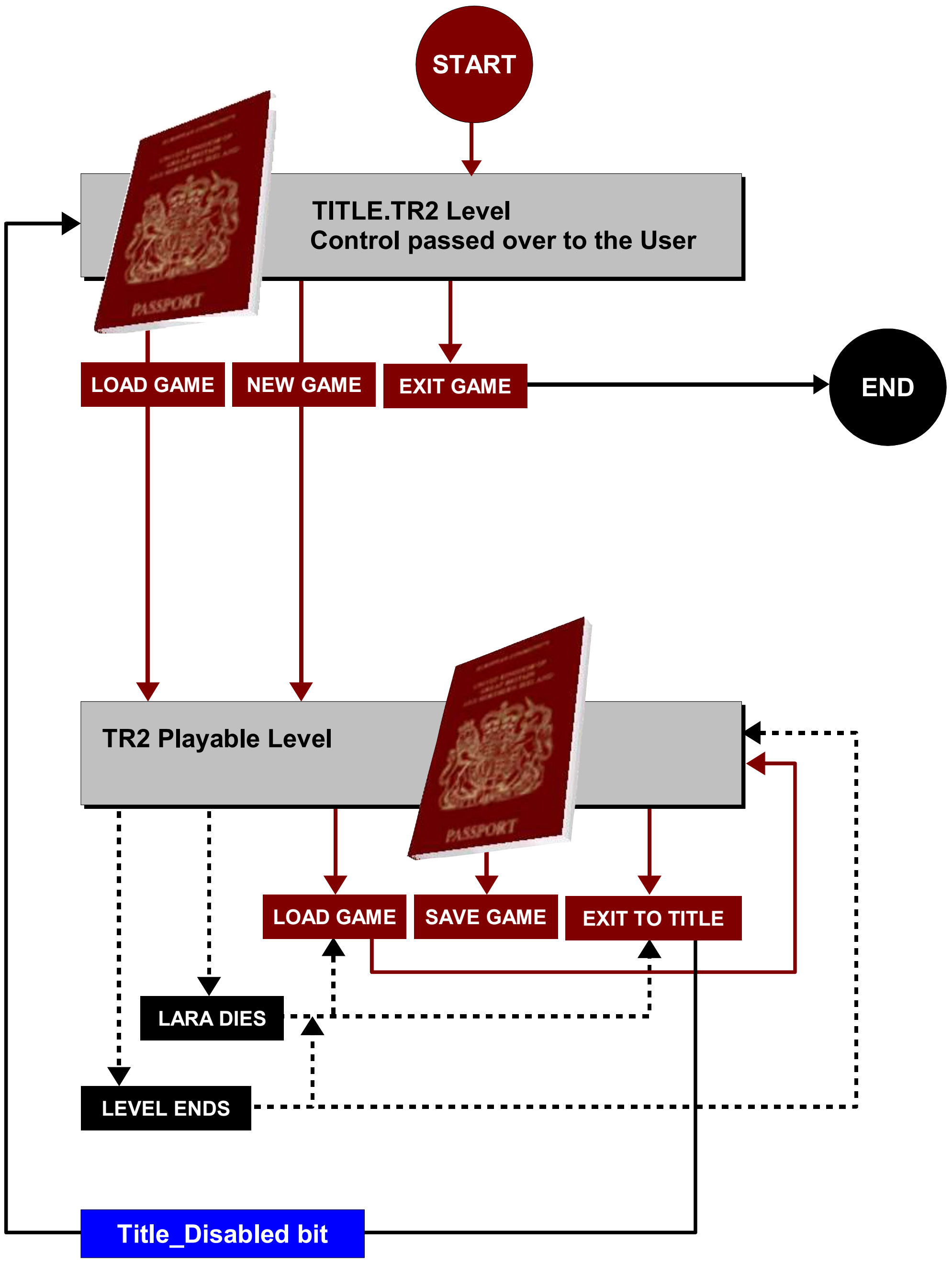
NOTES:

- The space between the Direction and the number that follows is optional.
- Usually the Directions yield nothing if their target does not exist, and an EXIT_TO_TITLE is issued out as a replacement.
- The Direction LEVEL %d is overridden by the **SingleLevel** field, which will force its own index.
- Meaningless Directions will be compiled as a LEVEL 1, and logged as STARTGAME 1.
- A decimal value entered instead of a Direction will produce **$FFFF FFFF** and will be logged with a **(null) 255** entry in the Script.LOG file. In this case an EXITGAME will happen.
- Fiddling with the EXIT_TO_TITLE requests may create unplayable situations. Better leave the **Title_Disabled** bit in the **Flags** field as 0, and leave the **Title_Replace** Direction as -1, which are the default values.

GAME FLOW CHART – PART 1



GAME FLOW CHART – PART 2



PART 1 of the flowchart shows most of the possibilities. The default path is the one on the left, where **FirstOption** = EXIT_TO_TITLE and where the **Title_Disabled** bit = 0. **PART 2** is simplified in order not to repeat the options if the **Title_Disabled** bit = 1. The blue box is there as a remainder, but the details were omitted.

Blank page

CHEATING ON A TR2 GAME[12]

To finish up a level:
Light up a flare.
Small step forward.
Small step backwards.
Turn around non-stop three times.
Jump forward.
The statistics panel shows up, the level ends.

To get all the weapons and maximize the ammo:
Light up a flare.
Small step forward.
Small step backwards.
Turn around non-stop three times.
Jump backwards.
There is a clicking sound that announces the update in the inventory. Now there is a Shotgun with 83 shells, the Automatic Pistols with 500 shells, the Uzis with 5000 shells, the M16 with 5000 shells, the Grenade Launcher with 5000 grenades, the Harpoon Gun with 5000 harpoons. It adds 50 Flares, 50 Small Medi Packs and 50 Large Medi Packs.

TR2 DEMO SEQUENCES

With **Tomb Raider II** Core Design implemented three demo sequences. If left alone in the Title Screen with no user input, the game cycles through the demos. If a demo sequence is interrupted by the user the game returns to the Title Screen ready to start a new game. This is useful to have it displaying in shops, advertising the game.

From the Title Screen, after 30 seconds of inactivity the game runs a 30 seconds demo based on “Venice”, where Lara breaks through the window of a shack to fight one of Bartoli's goons.
When the demo sequence ends the game goes back to the Title Screen.

After another 30 seconds of inactivity the game runs one more 30 seconds demo based on “Wreck of the Maria Doria”, where Lara fights a diver with harpoons.
When the demo sequence ends the game goes back to the Title Screen.

After another 30 seconds of inactivity the game runs one more 30 seconds demo based on “Tibetan Foothills”, where Lara fights a couple of goons and rides the snowmobile.
When the demo sequence ends the game goes back to the Title Screen.
And the process repeats.

These demo sequences are not activated while the game is being played.

The demo level distributions of **Tomb Raider II** do not run demo sequences.

[12] *From Stella's Tomb Raider site.*

TR2 FUNCTION KEYS

F1	Reduce graphics resolution.
F2	Increase graphics resolution.
Shift F1	Reduce colour depth.
Shift F2	Increase colour depth.
F3	Reduce screen size.
F4	Increase screen size.
F5	Save Game.
F6	Load Game.
F7	Toggle Z-buffering on / off.
F8	Toggle bilinear filtering on / off.
Shift F8	Toggle perspective correction on / off.
F11	Toggle dithering on / off.
F12	Toggle full screen / windowed.

Do not map the **S** key to any control in the game. Each time the **S** key is pressed, a screen shot will be saved to the **Tomb Raider II** directory. The file will be saved in targa format, and named from "tomb0000.tga", incrementally.

There is no hot key to exit **Tomb Raider II**. To exit the game, press ALT-F4.

OPCODES AND OPERANDS

Some OpCodes do not work at all on the PC. They may be PSX codes or they may be features still under development at the time the game was released. The unused OpCodes are **0**, **1**, **2**, **8** and **12**. The unknowns are **11** and **13**. The meaning of the OpCodes and their Operands was found through trial and error with custom TR2 test levels.

DISPLAYING FULL MOTION VIDEOS:

NAME:	FMV
OPCODE :	**3**
OPERAND:	ID of the FMV in the script.

Full Motion Videos are pre-rendered high-resolution animated scenes.
These scenes are distributed as a low-resolution RPL audio-video file, playable with the ESCAPE player. This format was used with TR1/2/3 games.

The Script.TXT file must declare the path of the video files, usually something like fmv\logo.rpl, a location relative to the Engine's location. An absolute location can be declared, like D:\TR2\fmv\logo.rpl, provided that the disk is the same as the Engine. The GAMEFLOW compiler will parse this entry properly. In any case the compiler builds a list of the video file names as it finds them in the script. The index of a file name in this list is used as the ID of the respective FMV in the compiled DAT file.

These videos can be included at will in different Sequences in the script.
More then one FMV can be included per Sequence.

- in the **FRONTEND** Sequence;
- in the **GYM** Sequence, before the training level itself. A video file inserted after the training level will not be played. When the level ends, an EXIT_TO_TITLE is called, bypassing the rest of the Sequence.
- in a **LEVEL** Sequence, before or after the game level itself;
- In a **DEMOLEVEL** Sequence, before the demo level itself, only. After the demo an **On_Demo_End** is called, bypassing the rest of the Sequence.

PLAYING A GAME LEVEL:

NAME:	GAME
OPCODE :	**4**
OPERAND:	ID of the level in the script.

This is the OpCode that launches a playable level in a **LEVEL** Sequence.

The Script.TXT file must declare the path of the level files, usually something like data\levelname.tr2, a location relative to the Engine's location. An absolute location can be declared instead, something like D:\TR2\data\levelname.tr2, provided that the disk is the same as the Engine. The GAMEFLOW compiler will parse this entry properly. In any case the compiler builds a list of the level file names as it finds them in the script. The index of a file name in this list is used as the ID of the respective level in the compiled DAT file.

The GAME OpCode for a level must be preceded by the other Opcodes related to the same level. These would typically be the soundtrack, supplies and items.
Any other OpCodes which affect the level must also be declared before the GAME command.

DISPLAYING CUT SCENES:

NAME:	CUT
OPCODE :	**5**
OPERAND:	ID of the FMV in the script.

Cut scenes are animated scenes rendered in real-time by the game's Engine, stored inside dedicated TR2 levels.

The Script.TXT file must declare the path of the cut scene level files, usually something like data\cut1.tr2, a location relative to the Engine's location. An absolute location can be declared instead, even a non-standard one, something like D:\TR2\cut\cut1.tr2, provided that the disk is the same as the Engine. The GAMEFLOW compiler will parse this entry properly. In any case the compiler builds a list of the cut scene level file names as it finds them in the script. The index of a file name in this list is used as the ID of the respective cut scene level in the compiled DAT file.

These cut scene levels use CD audio sound tracks stored in the CDROM, same as the game levels. Therefore a soundtrack must be declared in the script, preceding the CUT command, for them to have audio, same procedure as the game levels. However, there is a critical difference here: whereas the game levels will run even without the CD audio, the cut scenes will not display at all without their audio.
There is another problem involving the cut scenes' audio, *as explained further down in this text.*

These special levels can be included at will in different Sequences in the script.
More then one cut scene can be included per Sequence.

- in the **FRONTEND** Sequence.
- in the **GYM** Sequence, before the training level itself. A cut scene level file inserted after the training level will not be displayed. When the training level ends, an EXIT_TO_TITLE is called, bypassing the rest of the Sequence.
- in a **LEVEL** Sequence, before or after the game level itself.
- in a **DEMOLEVEL** Sequence, before the demo level itself, only. After the demo an **On_Demo_End** is called, bypassing the rest of the Sequence.

Although true, the statements above may not work properly.
There is a problem with the CD audio of the cut scenes versus the CD audio of the game levels. If a cut scene is displayed before a game (or demo) level, the cut scene audio keeps on playing forever and overrides the audio of the game level. A cut scene will not override another cut scene. There is no interference with FMV files.

Given this CD audio situation, cut scenes may not be inserted *before* other levels. The GYM Sequence, therefore, cannot properly play the training level after displaying a cut scene. The same applies to the LEVEL and DEMOLEVEL Sequences. They will not play their levels properly after displaying a cut scene.

LEVEL COMPLETION STATISTICS:

NAME:	COMPLETE
OPCODE :	**6**
OPERAND:	None.

Displays the end of the level statistics: Time Taken, Secrets Found, Kills, Ammo Used, Hits, Health Packs Used, Distance Travelled.

DISPLAYING DEMO SEQUENCES:

NAME:	DEMO
NAME:	PCDEMO
OPCODE :	**7**
OPERAND:	ID of the level in the script.

This is the OpCode that launches a recorded demo sequence in a **DEMOLEVEL** Sequence.

Recorded demo sequences are scenes rendered in real-time, whose animations are stored inside standard TR2 game levels, showing Lara in action inside those levels. The movements are previously recorded in real-time during game playing, then stored in the same level and made available for demo purposes.

The Script.TXT file must declare the path of the level files containing the demo sequences, usually something like data\levelname.tr2, a location relative to the Engine's location. An absolute location can be declared instead, something like D:\TR2\data\levelname.tr2, provided that the disk is the same as the Engine. The GAMEFLOW compiler will parse this entry properly. In any case the compiler builds a list of the level file names as it finds them in the script. The index of a file name in this list is used as the ID of the respective demo sequence level in the compiled DAT file.

CLOSING SEQUENCES:

NAME:	END
OPCODE :	**9**
OPERAND:	None.

Closes the **OPTIONS**, **TITLE**, **FRONTEND**, **GYM**, **LEVEL** or **DEMOLEVEL** Sequences in the Script.TXT file.

PLAYING CD AUDIO TRACKS:

NAME:	TRACK
NAME:	PCTRACK
OPCODE :	**10**
OPERAND:	ID of the CD audio track in the script.

CD audio tracks are stored in the game's CDROM in a separate area, other then the data files area. These audio tracks are playable as any other audio CDROM disks. The index of the audio track is used as the ID of the respective soundtrack in the compiled DAT file.

These tracks contain the music playing in the background throughout the game. Or the dialogues for the cut scenes. Or the explanations for the training level.
The TRACK or PCTRACK command is used to select the background music or ambiance for the level.

This command must be declared in the Script.TXT before the level it relates to, be it a GAME, CUT or DEMO.

LARA WITHOUT PISTOLS:

NAME:	REMOVE_WEAPONS
OPCODE :	**14**
OPERAND:	None.

Lara starts the level without Pistols or any other weapons.
This command can be followed by the STARTINV command which can be used to give some weapons back to Lara. In that case Lara will start the level with those weapons, but still with no Pistols. Lara can have the Pistols back during the game if she finds them and picks them up. The Pistols will then be available again in the Inventory and Lara can select and use them.

FINISHING THE GAME:

NAME:	GAMECOMPLETE
OPCODE :	**15**
OPERAND:	None.

This OpCode will terminate the game. The final statistics will be displayed, showing the sum of the individual level statistics. The final sequence of PCX files picturing the credits will be launched, the musical score ID = 52 will be played.

ANGLE FOR THE CUT CAMERA:

NAME:	CUTANGLE
OPCODE :	**16**
OPERAND:	Horizontal rotation of the camera, clockwise.

The animation of the camera of a cut scene is rotated by the given angle. The 3D animated characters will rotate together with the camera, Lara included. The elements placed by the Room Editor itself are not affected by this rotation.

This is used to match the North-South orientation of the Room Editor and the North-South orientation of the 3D animated characters originated from a CAD application.

The final result is identical to rotating the level itself, counter-clockwise.

Angle = 360 * Operand / 65536

Operand = Angle * 65536 / 360

As a quick reference, here are some values:

0º	=	$0000	=	0
45º	=	$2000	=	8192
90º	=	$4000	=	16384
135º	=	$6000	=	24576
180º	=	$8000	=	32768
225º	=	$A000	=	40960
270º	=	$C000	=	49152

DEATH BY DEPTH:

NAME:	NOFLOOR
OPCODE :	**17**
OPERAND:	Depth.

If walking down, Lara dies when her feet reach the given Depth.
If falling down, 4 to 5 extra blocks are added to Depth.
Each vertical block corresponds to 1024 units.
Depth is measured relative to the floor of the room where Lara is placed in the beginning of the game. We will call it “ground-zero” for short.

Depth = NumBlocks * 1024

NumBlocks = Depth / 1024

If Depth = 0 then the feature is disabled.
If Depth = 1 then Lara cannot even step on the portal, that's sudden death.
If Depth = 256 then Lara can walk down the equivalent to ¼ of a block.

As a quick reference, here are some values (Depth versus blocks):

256	=	¼
512	=	½
768	=	¾
1024	=	1
2048	=	2
3072	=	3
4096	=	4

When Lara walks down through stacked rooms, the depth of her feet below ground-zero will increase until Depth is reached. The level will end there.
The same applies to water rooms. Deeper then a given depth below ground-zero, Lara dies of sudden death.

The Death-by-Depth feature has no associated animations. Lara dies suddenly and the game jumps out to the Option Ring. **Tomb Raider II** uses this feature in the level “Floating Islands”, where Lara falls into the blackness of the depths below, in fact disappearing, and the level exits suddenly.

GIVING ITEMS TO LARA:

NAME:	STARTINV
NAME:	BONUS
OPCODE :	**18**
OPERAND:	ID of the item.

This OpCode is used to give items to Lara, like weapons, ammunition, supplies or puzzles. There are two names for this same command. STARTINV gives the items when the Level starts, BONUS gives the items when all the secrets are found. Because they use the same OpCode, **18**, the difference is established by the ID of the item: the Operand for BONUS is the same as STARTINV plus **1000**.

STARTING WITH A SPECIAL ANIMATION:

NAME:	STARTANIM
OPCODE :	**19**
OPERAND:	ID of the animation.

This OpCode makes the level start with a special animation performed by Lara. Examples can be seen in **Tomb Raider II** in the "Offshore Rig" level, where Lara is laying down on the floor, awakes and stands up. Or in the "Home Sweet Home" level, where Lara is sitting down on her bed watching the Dagger of Xian.

ACCOUNTING SECRETS:

NAME:	SECRETS
OPCODE :	**20**
OPERAND:	OnOff status.

All the playable levels in **Tomb Raider II** are supposed to have three secrets. Therefore, the total number of secrets in the game is supposed to be given by the equation `TotalSecrets = 3 * NumLevels`. But it may be different.
In fact, it is also possible that a certain level has no secrets at all.
In that case such level should not be accounted in the equation above.

The purpose of SECRETS is to signal if a level is to be accounted or not.
If the Operand is **0**, the level is not accounted for secrets – even if it has any.
A non-zero value means that the level must be accounted for secrets. The value itself has no meaning other then that.

KILL ALL ENEMIES TO COMPLETE:

NAME:	KILLTOCOMPLETE
OPCODE :	**21**
OPERAND:	None.

All the enemies in the level must be killed, for the level to exit as complete.
An example in **Tomb Raider II** is the "Home Sweet Home" level.

LARA WITHOUT AMMUNITION:

NAME:	REMOVE_AMMO
OPCODE :	**22**
OPERAND:	None.

Lara starts the level without ammunition or medi packs.

RECREATING THE SCRIPT OF "DAGGER OF XIAN"

This is not the original text of the original script. This is a recreation. The TOMBPC.DAT file that was compiled from this script, using Core Design's GAMEFLOW utility, is equal to the original DAT file. This proves the consistency of this research, despite of the fact that some fields are still listed as unknown. All-in-all, this script is a good reference for the development of custom scripts for the TR2 Series of custom levels.

```
//==========================================================================
//==========================================================================
//==========================================================================
//
//       TOMB RAIDER II SCRIPT
//
//       Recreation of a script for "DAGGER OF XIAN".
//       To be compiled with Core Design's GAMEFLOW.EXE utility.
//
//       This script was not produced and is not supported or endorsed
//       by Eidos or Core Design.
//
//       Research by IceBerg, August/2005.
//
//==========================================================================
//==========================================================================
//==========================================================================


DESCRIPTION:    Tomb Raider II Script. Final Release Version 1.1 (c) Core Design Ltd 1997


//--------------------------------------------------------------------------
//
//              O P T I O N S
//
//--------------------------------------------------------------------------


OPTIONS:

   CYPHER_CODE:        166

   SECRET_TRACK:       47

END:


//--------------------------------------------------------------------------
//
//              T I T L E
//
//--------------------------------------------------------------------------


TITLE:

   GAME:       data\title.tr2         // Title Ring

   PCFILE:     data\title.pcx
   PCFILE:     data\legal.pcx
   PCFILE:     data\titleUS.pcx
   PCFILE:     data\legalUS.pcx
   PCFILE:     data\titleJAP.pcx
   PCFILE:     data\legalJAP.pcx

   TRACK:      64

END:
```

```
//--------------------------------------------------------------------------
//
//              F R O N T E N D
//
//--------------------------------------------------------------------------


FRONTEND:                            // Full Motion Videos

   FMV:         fmv\logo.rpl
   FMV:         fmv\ancient.rpl

END:


//--------------------------------------------------------------------------
//
//              G Y M
//
//--------------------------------------------------------------------------


GYM:            Lara's Home          // Level 0

   SECRETS:     0
   TRACK:       0
   GAME:        data\assault.tr2

END:


//--------------------------------------------------------------------------
//
//              L E V E L S
//
//--------------------------------------------------------------------------


LEVEL:          The Great Wall       // Level 1

   FMV:         fmv\modern.rpl

   TRACK:       33
   BONUS:       ROCKET
   BONUS:       ROCKET_AMMO
   BONUS:       ROCKET_AMMO
   BONUS:       MEDI
   KEY1:        Guardhouse Key
   KEY2:        Rusty Key
   GAME:        data\wall.tr2

   TRACK:       3
   CUTANGLE:    0
   CUT:         data\cut1.TR2

   COMPLETE:

END:

//--------------------------------------------------

LEVEL:          Venice               // Level 2

   TRACK:       0
   BONUS:       AUTOPISTOLS_AMMO
   BONUS:       AUTOPISTOLS_AMMO
   BONUS:       AUTOPISTOLS_AMMO
   BONUS:       AUTOPISTOLS_AMMO
   KEY1:        Boathouse Key
   KEY2:        Steel Key
   KEY3:        Iron Key
   GAME:        data\boat.TR2

   COMPLETE:

END:
```

```
//-------------------------------------------------

LEVEL:         Bartoli's Hideout      // Level 3

   TRACK:      0
   BONUS:      SHOTGUN_AMMO
   BONUS:      SHOTGUN_AMMO
   BONUS:      SHOTGUN_AMMO
   BONUS:      SHOTGUN_AMMO
   SUNSET:
   KEY1:       Library Key
   KEY2:       Detonator Key
   GAME:       data\venice.TR2

   COMPLETE:

END:

//-------------------------------------------------

LEVEL:         Opera House            // Level 4

   TRACK:      31
   BONUS:      UZIS
   BONUS:      UZI_AMMO
   BONUS:      UZI_AMMO
   BONUS:      UZI_AMMO
   BONUS:      UZI_AMMO
   PUZZLE1:    Relay Box
   PUZZLE2:    Circuit Board
   KEY1:       Ornate Key
   GAME:       data\opera.TR2

   TRACK:      4
   CUT:        data\cut2.TR2

   COMPLETE:

END:

//-------------------------------------------------

LEVEL:         Offshore Rig           // Level 5

   FMV:        fmv\landing.rpl

   TRACK:      58
   STARTANIM:  8
   REMOVE_WEAPONS:
   BONUS:      UZIS
   BONUS:      UZI_AMMO
   BONUS:      UZI_AMMO
   KEY1:       Red Pass Card
   KEY2:       Yellow Pass Card
   KEY3:       Green Pass Card
   GAME:       data\rig.TR2

   COMPLETE:

END:

//-------------------------------------------------

LEVEL:         Diving Area            // Level 6

   TRACK:      58
   BONUS:      UZI_AMMO
   BONUS:      UZI_AMMO
   BONUS:      UZI_AMMO
   BONUS:      UZI_AMMO
   PUZZLE1:    Machine Chip
   KEY1:       Red Pass Card
   KEY4:       Blue Pass Card
   GAME:       data\platform.TR2

   TRACK:      5
   CUT:        data\cut3.TR2
   COMPLETE:

END:

//--------------------------------------------------

LEVEL:          40 Fathoms            // Level 7

   FMV:         fmv\ms.rpl

   TRACK:       34
   BONUS:       HARPOON_AMMO
   BONUS:       HARPOON_AMMO
   BONUS:       HARPOON_AMMO
   BONUS:       HARPOON_AMMO
   GAME:        data\unwater.TR2

   COMPLETE:

END:

//--------------------------------------------------

LEVEL:          Wreck of the Maria Doria // Level 8

   TRACK:       31
   BONUS:       ROCKET
   BONUS:       ROCKET_AMMO
   BONUS:       ROCKET_AMMO
   PUZZLE1:     Circuit Breaker
   KEY1:        Rest Room Key
   KEY2:        Rusty Key
   KEY3:        Cabin Key
   GAME:        data\keel.TR2

   COMPLETE:

END:

//--------------------------------------------------

LEVEL:          Living Quarters       // Level 9

   TRACK:       34
   BONUS:       M16_AMMO
   BONUS:       M16_AMMO
   BONUS:       M16_AMMO
   BONUS:       M16_AMMO
   KEY1:        Theatre Key
   KEY2:        Rusty Key
   GAME:        data\living.TR2

   COMPLETE:

END:

//--------------------------------------------------

LEVEL:          The Deck              // Level 10

   TRACK:       31
   BONUS:       ROCKET_AMMO
   BONUS:       ROCKET_AMMO
   BONUS:       ROCKET_AMMO
   BONUS:       ROCKET_AMMO
   PUZZLE4:     The Seraph
   KEY2:        Stern Key
   KEY3:        Storage Key
   KEY4:        Cabin Key
   GAME:        data\deck.TR2

   COMPLETE:

END:
```

```
//---------------------------------------------------

LEVEL:         Tibetan Foothills      // Level 11

   FMV:        fmv\crash.rpl

   TRACK:      33
   STARTINV:   PUZZLE4
   BONUS:      UZI_AMMO
   BONUS:      UZI_AMMO
   BONUS:      UZI_AMMO
   BONUS:      UZI_AMMO
   PUZZLE4:    The Seraph
   KEY1:       Drawbridge Key
   KEY2:       Hut Key
   GAME:       data\skidoo.TR2

   COMPLETE:

END:

//---------------------------------------------------

LEVEL:         Barkhang Monastery     // Level 12

   TRACK:      0
   STARTINV:   PUZZLE4
   BONUS:      M16_AMMO
   BONUS:      M16_AMMO
   BONUS:      M16_AMMO
   BONUS:      M16_AMMO
   PUZZLE1:    Prayer Wheels
   PUZZLE2:    Gemstones
   PUZZLE4:    The Seraph
   KEY1:       Strongroom Key
   KEY2:       Trapdoor Key
   KEY3:       Rooftops Key
   KEY4:       Main Hall Key
   GAME:       data\monastry.TR2

   COMPLETE:

END:

//---------------------------------------------------

LEVEL:         Catacombs of the Talion // Level 13

   TRACK:      31
   BONUS:      ROCKET_AMMO
   BONUS:      ROCKET_AMMO
   BONUS:      M16_AMMO
   BONUS:      M16_AMMO
   PUZZLE1:    Tibetan Mask
   PICKUP1:    Gong Hammer
   GAME:       data\catacomb.TR2

   COMPLETE:

END:

//---------------------------------------------------

LEVEL:         Ice Palace             // Level 14

   TRACK:      31
   DEADLY_WATER:
   BONUS:      ROCKET_AMMO
   BONUS:      ROCKET_AMMO
   BONUS:      ROCKET_AMMO
   BONUS:      ROCKET_AMMO
   PUZZLE1:    Tibetan Mask
   PICKUP2:    Talion
   KEY2:       Gong Hammer
   GAME:       data\icecave.TR2

   COMPLETE:
```

```
END:

//--------------------------------------------------

LEVEL:          Temple of Xian        // Level 15

   FMV:         fmv\jeep.rpl

   TRACK:       59
   BONUS:       UZI_AMMO
   BONUS:       UZI_AMMO
   BONUS:       UZI_AMMO
   BONUS:       UZI_AMMO
   BONUS:       UZI_AMMO
   BONUS:       UZI_AMMO
   BONUS:       UZI_AMMO
   BONUS:       UZI_AMMO
   PUZZLE1:     The Dragon Seal
   KEY2:        Gold Key
   KEY3:        Silver Key
   KEY4:        Main Chamber Key
   GAME:        data\emprtomb.TR2

   TRACK:       30
   CUTANGLE:    0
   CUT:         data\cut4.tr2

   COMPLETE:

END:

//--------------------------------------------------

LEVEL:          Floating Islands      // Level 16

   TRACK:       59
   NOFLOOR:     9728    // $2600
   BONUS:       ROCKET_AMMO
   BONUS:       ROCKET_AMMO
   BONUS:       ROCKET_AMMO
   BONUS:       ROCKET_AMMO
   BONUS:       ROCKET_AMMO
   BONUS:       ROCKET_AMMO
   BONUS:       ROCKET_AMMO
   BONUS:       ROCKET_AMMO
   PUZZLE1:     Mystic Plaque
   PUZZLE2:     Mystic Plaque
   GAME:        data\floating.TR2

   COMPLETE:

END:

//--------------------------------------------------

LEVEL:          The Dragon's Lair     // Level 17

   SECRETS:     0
   TRACK:       59
   PUZZLE1:     Mystic Plaque
   PUZZLE2:     Dagger of Xian
   GAME:        data\xian.TR2

   COMPLETE:

   FMV:         fmv\end.rpl

END:
```

```
//-------------------------------------------------------------------------
//
//             B O N U S   L E V E L
//
//-------------------------------------------------------------------------


LEVEL:         Home Sweet Home        // Level 18

   SECRETS:    0
   STARTINV:   KEY1
   STARTANIM:  9
   KILLTOCOMPLETE:
   TRACK:      0
   REMOVE_WEAPONS:
   REMOVE_AMMO:
   PUZZLE1:    Dagger of Xian
   KEY1:       Gun Cupboard Key
   GAME:       data\house.TR2

   TRACK:      52
   GAMECOMPLETE:

END:


//-------------------------------------------------------------------------
//
//             D E M O   L E V E L S
//
//-------------------------------------------------------------------------


DEMOLEVEL:     Venice                 // Level 19

   TRACK:      0
   KEY1:       Boathouse Key
   KEY2:       Steel Key
   KEY3:       Iron Key
   DEMO:       data\boat.TR2

END:

//--------------------------------------------------

DEMOLEVEL:     Wreck of the Maria Doria //Level 20

   TRACK:      32
   PUZZLE1:    Circuit Breaker
   KEY1:       Rest Room Key
   KEY2:       Rusty Key
   KEY3:       Cabin Key
   DEMO:       data\keel.TR2

END:

//--------------------------------------------------

DEMOLEVEL:     Tibetan Foothills      // Level 21

   TRACK:      33
   STARTINV:   PUZZLE4
   PUZZLE4:    The Seraph
   KEY1:       Drawbridge Key
   KEY2:       Hut Key
   DEMO:       data\skidoo.TR2

END:


//=========================================================================
//=========================================================================

GameStrings:   English.txt

//=========================================================================
//=========================================================================
```

The language file that goes with this script text is the English version. The number of strings is fixed. A different number will make the compiler exit with an error message. There must be a total of **89** GameStrings, **41** PCStrings and **80** PSXStrings. The unused strings must be present anyway. A dummy string like “spare” is used as a filler.
This ENGLISH.TXT file also serves the purposes of a **language template for custom levels**.
The strings contained herein are game-specific, not level-specific. The level-specific strings are declared in the script text file, not in the language text file.

```
//==============================================
//==============================================
//==============================================
//
//      PC & PSX VERSION OF GENERAL GAME STRINGS
//
//      Recreation of
//      Tomb Raider II strings - English
//
//      This script was not produced and is not
//      supported or endorsed by
//      Eidos or Core Design.
//
//      Research by IceBerg, August/2005.
//
//==============================================
//==============================================
//==============================================


//----------------------------------------------
//      GAME STRINGS (89 entries)
//----------------------------------------------

GAME_STRINGS:

INVENTORY
OPTION
ITEMS
GAME OVER
Load Game
Save Game
New Game
Restart Level
Exit to Title
Exit Demo
Exit Game
Select Level
Save Position
Select Detail
High
Medium
Low
Walk
Roll
Run
Left
Right
Back
Step Left
?
Step Right
?
Look
Jump
Action
Draw Weapon
?
Inventory
Flare
Step
Statistics
Pistols
Shotgun
```

```
Automatic Pistols
Uzis
Harpoon Gun
M16
Grenade Launcher
Flare
Pistol Clips
Shotgun Shells
Automatic Pistol Clips
Uzi Clips
Harpoons
M16 Clips
Grenades
Small Medi Pack
Large Medi Pack
Pickup
Puzzle
Key
Game
Lara's Home
LOADING
Time Taken
Secrets Found
Location
Kills
Ammo Used
Hits
Saves Performed
Distance Travelled
Health Packs Used
Release Version 1.1
None
Finish
BEST TIMES
No Times Set
N/A
Current Position
Final Statistics
of
Story so far...
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare

END:


//----------------------------------------------
//      PC STRINGS (41 entries)
//----------------------------------------------

PC_STRINGS:

Detail Levels
Demo Mode
Sound
Controls
Gamma
Set Volumes
User Keys
The file could not be saved!
Try Again?
YES
NO
Save Complete!
No save games!
None valid
Save Game?
- EMPTY SLOT -
OFF
```

```
ON
Setup Sound Card
Default Keys
DOZY
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare

END:


//-----------------------------------------------
//      PLAYSTATION STRINGS (80 entries)
//-----------------------------------------------

PSX_STRINGS:

spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
```

```
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare
spare

END:

//==============================================
//==============================================
//==============================================
```

RECREATING THE SCRIPT OF “THE GOLDEN MASK”

This is not the original text of the original script. This is a recreation. The TOMBPC.DAT file that was compiled from this script, using Core Design's GAMEFLOW utility, is equal to the original DAT file. This proves the consistency of this research, despite of the fact that some fields are still listed as unknown. All-in-all, this script is a good reference for the development of custom scripts for the TR2 Series of custom levels.

```
//=======================================================================
//=======================================================================
//=======================================================================
//
//      TOMB RAIDER II SCRIPT
//
//      Recreation of a script for "THE GOLDEN MASK".
//      To be compiled with Core Design's GAMEFLOW.EXE utility.
//
//      This script was not produced and is not supported or endorsed
//      by Eidos or Core Design.
//
//      Research by IceBerg, August/2005.
//
//=======================================================================
//=======================================================================
//=======================================================================


DESCRIPTION:    Tomb Raider II Script. Final Release Version 1.1 (c) Core Design Ltd 1997


//-----------------------------------------------------------------------
//
//            O P T I O N S
//
//-----------------------------------------------------------------------


OPTIONS:

   CYPHER_CODE:        166

   SECRET_TRACK:       47

END:


//-----------------------------------------------------------------------
//
//            T I T L E
//
//-----------------------------------------------------------------------


TITLE:

   GAME:       data\title.tr2         // Title Ring

   PCFILE:     data\title.pcx
   PCFILE:     data\legal.pcx
   PCFILE:     data\titleUS.pcx
   PCFILE:     data\legalUS.pcx
   PCFILE:     data\titleJAP.pcx
   PCFILE:     data\legalJAP.pcx

   TRACK:      64

END:
```

```
//---------------------------------------------------------------------------
//
//              F R O N T E N D
//
//---------------------------------------------------------------------------


FRONTEND:                           // Full Motion Videos

   FMV:         fmv\logo.rpl

END:


//---------------------------------------------------------------------------
//
//              G Y M
//
//---------------------------------------------------------------------------
// NOTE: This is declared in the original TOMBPC.DAT, but the game itself
// does not have this file. The Title Screen does not have any option in
// the Menu Ring to access any training level.



GYM:            Lara's Home         // Level 0

   SECRETS:     0
   TRACK:       0
   GAME:        data\assault.tr2

END:


//---------------------------------------------------------------------------
//
//              L E V E L S
//
//---------------------------------------------------------------------------


LEVEL:          The Cold War        // Level 1

   KEY1:        Guardroom Key
   KEY2:        Shaft 'B' Key
   TRACK:       33
   GAME:        data\level1.TR2

   COMPLETE:

END:

//--------------------------------------------------

LEVEL:          Fool's Gold         // Level 2

   PUZZLE1:     Circuit Board
   KEY1:        CardKey 1
   KEY2:        CardKey 2
   TRACK:       58
   GAME:        data\level2.TR2

   COMPLETE:

END:

//--------------------------------------------------

LEVEL:          Furnace of the Gods   // Level 3

   PUZZLE1:     Mask of Tornarsuk
   PUZZLE2:     Gold Nugget
   TRACK:       59
   GAME:        data\level3.TR2

   COMPLETE:

END:
```

```
//---------------------------------------------------

LEVEL:         Kingdom               // Level 4

   PUZZLE1:    Mask of Tornarsuk
   STARTINV:   PUZZLE1
   TRACK:      31
   GAME:       data\level4.TR2

   GAMECOMPLETE:

END:


//--------------------------------------------------------------------------
//
//             B O N U S   L E V E L
//
//--------------------------------------------------------------------------


LEVEL:         Nightmare In Vegas    // Level 5

   PUZZLE1:    Elevator Junction
   PUZZLE2:    Door Circuit
   KEY1:       Hotel Key
   TRACK:      34
   GAME:       data\level5.TR2

   COMPLETE:

END:

//==========================================================================
//==========================================================================
//==========================================================================

GameStrings:   English.txt

//==========================================================================
//==========================================================================
//==========================================================================
```

GAMEFLOW.EXE

Blank page

TR2 – THE STRINGS

Looking into the compiler EXE file with an ordinary text editor, we can find many strings that the compiler expects to find in the script text, together with strings used by the compiler itself:

data\venice.dem

data\cut4.cut
data\cut3.cut
data\cut2.cut
data\cut1.cut

data\house.tr2
data\floating.tr2
data\emprtomb.tr2
data\icecave.tr2
data\catacomb.tr2
data\monastry.tr2
data\skidoo.tr2
data\dock.tr2
data\living.tr2
data\keel.tr2
data\unwater.tr2
data\rig.tr2
data\opera.tr2
data\venice.tr2
data\boat.tr2
data\wall.tr2
data\test.tr2
data\assault.tr2

fmv\fendseq.rpl
fmv\jeep.rpl
fmv\seaplane.rpl
fmv\minisub.rpl
fmv\rig.rpl
fmv\wallpres.rpl
fmv\wallold.rpl
fmv\home.rpl
fmv\escape.rpl
fmv\corelogo.rpl

data\title.tr2

data\titleh.pcx

data\eidospc

The fact that these strings are stored in the compiler itself does not, however, prevent it from using custom strings. In fact, the compiler will use the level names and file names supplied in the script text.
There is a large collection of strings to be found in the remaining of the GAMEFLOW compiler:
This is what the compiler expects to find as input, or how it expresses itself in the output. Colours are used where already tested. Where relevant, // comments were included.

DISABLED
ENABLED
OFF
ON

KEY4	// id = 26
KEY3	// id = 25
KEY2	// id = 24
KEY1	// id = 23
PUZZLE4	// id = 22
PUZZLE3	// id = 21
PUZZLE2	// id = 20
PUZZLE1	// id = 19
PICKUP2	// id = 18
PICKUP1	// id = 17
BIGMEDI	// id = 16
MEDI	// id = 15

FLARES	// id =14
ROCKET_AMMO	// id = 13
M16_AMMO	// id = 12
HARPOON_AMMO	// id = 11
UZI_AMMO	// id = 10
AUTOPISTOLS_AMMO	// id = 9
SHOTGUN_AMMO	// id = 8
PISTOLS_AMMO	// id = 7
ROCKET	// id = 6
M16	// id = 5
HARPOON	// id = 4
UZIS	// id = 3
AUTOPISTOLS	// id = 2
SHOTGUN	// id = 1
PISTOLS	// id = 0

```
TITLE_DESELECT     //  $0900
EXIT_TO_OPTION     //  $0800
EXITGAME           //  $0700
LEVELCOMPLETE      //  $0600
EXIT_TO_TITLE      //  $0500
STARTDEMO          //  $0400
STARTFMV           //  $0300
STARTCINE          //  $0200
STARTSAVEDGAME     //  $0100
STARTGAME          //  $0000

scripts\pcfinal.txt

tombPC.dat

gameflow.log

strings.txt

scripts\PCstrOUT.txt

Invalid option
ERROR: Too many parameters
Lockout = %d

Key4 Strings
Key4 Strings
Key3 Strings
Key3 Strings
Key2 Strings
Key2 Strings
Key1 Strings
Key1 Strings

Pickup2 Strings
Pickup2 Strings
Pickup1 Strings
Pickup1 Strings

Puzzle4 Strings
Puzzle4 Strings
Puzzle3 Strings
Puzzle3 Strings
Puzzle2 Strings
Puzzle2 Strings
Puzzle1 Strings
Puzzle1 Strings

PC Strings
PC Strings

Game Strings
Game Strings

None    %d

Writing Demo Level Numbers
Demo Levels:
Writing Demo Level Numbers
```

```
Writing Level Sequences
Writing Level Sequences

Writing FrontEnd Sequence
Writing FrontEnd Sequence

Total Sequence Data Size= %d bytes
Total Sequence Data Size= %d bytes

LEVEL%d SEQ (SIZE: %d bytes)
FRONTEND SEQ (SIZE: %d bytes)

Cutscene FileNames
Cutscene FileNames

Level FileNames
Level FileNames

FMV FileNames
FMV FileNames

Title FileNames
Title FileNames

Picture FileNames
Picture FileNames

Level Names
Level Names

dozy_cheat:              %s
lockout_optionring:      %s
screensizing:            %s
loadsave:                %s
cheatmodecheck:          %s
NoInput_time:            %d
noinput_timeout:         %s
on_demo_end:             %s %d
on_demo_interrupt:       %s %d
ondeath_ingame:          %s %d
ondeath demo_mode:       %s %d
TITLE REPLACED WITH:     %s %d
TITLE IS DISABLED
firstOption:             %s %d
Demo Options;-

cypher_code:             %d
TitleFiles:              %d
Title Track:             %d
Pictures:                %d
Cutscenes:               %d
FMV:                     %d
Demos:                   %d
Levels:                  %d
Gameflow Options;-

sizeof GAMEFLOW_INFO =   %d
Script Version: %d
Date Produced: %s %s
14:40:07    Oct 30 1997
```

```
Gameflow script produced for;
Producing gameflow script for;
ERROR: Couldn't open '%s'
ERROR: Couldn't open log file '%s'
str
log
txt
ERROR: Unknown sequence command (%d)
ERROR: No sequence!!!

ENDSEQUENCE
KILL TO COMPLETE
REMOVE_AMMO
REMOVE_WEAPONS
START INVENTORY:    %s
ADD TO INVENTORY:   %s
NUM SECRETS:        %d
LARA START ANIM:    %d
NO FLOOR:           %d
DEADLY WATER
LOADING PIC:        %s
SUNSET ENABLED
SETTRACK:           %d
JUMPTOSEQ:          %d
DEMOPLAY:           %s
GAMECOMPLETE
LEVCOMPLETE
CUT ANGLE:          0x%x
CUTSCENE:           %s
STARTLEVEL:
STARTLEVEL:
FMV:                %s
LIST END <<<<
LIST START>>>
PICTURE:            %s
Buffer size =       %d
Strings =           %d

ERROR: Cannot open '%s' info
Usage:  pcscript [/i] script.txt [strings.txt]
GameFlow Script Converter (PC Version)
data\out.dat
successfully
ERROR: Incorrect number of psx strings.
ERROR: Incorrect number of pc strings.
 %d:'%s' %d strings, there should be %d
ERROR: Incorrect number of game strings.
ERROR: Unknown command '%s %s'
Reading GameStrings...
GAMESTRINGS
DEMOLEVEL %d: %s
DEMOLEVEL
LEVEL
LEVEL %d: %s
LEVEL
GYM
GYM: %s
GYM FRONTEND SEQUENCE
FRONTEND SEQUENCE
FRONTEND
```

```
TITLE_FILES
TITLE FILES
TITLE
OPTIONS
OPTIONS
OPTIONS DESCRIPTION
DESCRIPTION
Script;-
Creating script...
Could not open script file '%s' r Done =  Found
kanji string - '%s'
ERROR: '*/' without '/*'

JAPANESE      // id = 4
AMERICAN      // id = 3
German        // id = 2
FRENCH        // id = 1
ENGLISH       // id = 0

ERROR: Unknown option '%s'

END                         // opcode = 9
ENABLE_CHEAT_CODE                       // bit[11]
SELECT_ANY_LEVEL                        // bit[10]
CYPHER_CODE
DOZY_CHEAT_ENABLED                      // bit[7]
LOCKOUT_OPTIONRING                      // bit[6]
SCREENSIZING_DISABLED                   // bit[5]
LOADSAVE_DISABLED                       // bit[4]
NOINPUT_TIMEOUT                         // bit[3]
CHEATMODECHECK_DISABLED                 // bit[2]
TITLE_DISABLED                          // bit[1]
DEMOVERSION                             // bit[0]
SINGLELEVEL
ON_DEMO_END
ON_DEMO_INTERRUPT
NOINPUT_TIME
ONDEATH_INGAME
ONDEATH_DEMO_MODE
TITLE_REPLACE
FIRSTOPTION
SECRET_TRACK
LANGUAGE
SEQUENCE
DEMO                        // opcode = 7
WARNING: EOF before end of sequence
ERROR: Unkown command '%s %s'
SPECIAL2
SPECIAL1
SECRET4
SECRET3
SECRET2
SECRET1

START INVENTORY: %s
ERROR: Could not match '%s' with inv items
STARTINV                    // opcode = 18
SECRET BONUS:               %s
BONUS                       // opcode = 18
```

```
KILL TO COMPLETE
KILLTOCOMPLETE      // opcode = 21
REMOVE AMMO
REMOVE_AMMO         // opcode = 22
REMOVE WEAPONS
REMOVE_WEAPONS      // opcode = 14
DEADLY WATER
DEADLY_WATER        // opcode = 13
LEVEL SECRETS:      %d
SECRETS             // opcode = 20
LARA START ANIM:    %d
STARTANIM           // opcode = 19
NO FLOOR ENABLED
NOFLOOR             // opcode = 17
SUNSET ENABLED
SUNSET              // opcode = 11
PSXTRACK
TRACK:              %d
PCTRACK             // opcode = 10
TRACK               // opcode = 10
GAMECOMPLETE
GAMECOMPLETE        // opcode = 15
COMPLETE
COMPLETE            // opcode = 6
PICTURE:            '%s'
PICTURE             // opcode = 0
LOAD_PIC            // opcode = 12
CUT:                '%s'
CUT                 // opcode = 5
CUTANGLE            // opcode = 16
PSXDEMO
PCDEMO:             '%s'
PCDEMO              // opcode = 7
DEMO:               '%s'
GAME:               '%s'
TR2
GAME                // opcode = 4
FMV_END
FMV_START
PSXFMV
PCFMV               // opcode = 3
FMV:                '%s'
RPL
FMV                 // opcode = 3
```

```
WARNING: EOF before end of title file list
Title Track:                %d
PSXFILE
PCFILE
PSX_STRINGS
PC_STRINGS
GAME_STRINGS
ERROR: Could not open string file '%s'

Tomb Raider II
PC Internal Development Version (c)
Core Design Ltd 1997
```

TR2 – THE SCRIPT LOG

The output of GAMEFLOW is logged into a file, PCFinal.LOG which, in the published German version of Tomb Raider II Golden Mask demo, looks like this (TABs readjusted for clarity) (-- comment separators added for clarity):

```
//------------------------------------------------- start of the LOG file.

Script;-
DESCRIPTION
          OPTIONS
          TITLE_FILES
          FRONTEND SEQUENCE
          GYM
          LEVEL
          LEVEL
          LEVEL
          LEVEL
          LEVEL
Script read successfully

Gameflow script produced for;-

Tomb Raider II Script. Final Release Version 1.1 (c) Core Design Ltd 1997.

Date Produced: 14:40:07 Oct 30 1997

Script Version: 3

sizeof GAMEFLOW_INFO = 128

//------------------------------------------------- Gameflow Options.

Gameflow Options;-
Levels:                 6
Demos:                  0
FMV:                    1
Cutscenes:              0
Pictures:               0
Title Track:            64
TitleFiles:             7
cypher_code:            166

//------------------------------------------------- Demo Options.

Demo Options;-
firstOption:            EXIT_TO_TITLE       0
ondeath demo_mode:      EXIT_TO_TITLE       0
ondeath_ingame:         STARTGAME           0
on_demo_interrupt:      EXIT_TO_TITLE       0
on_demo_end:            EXIT_TO_TITLE       0
noinput_timeout:        ON
NoInput_time:           900
cheatmodecheck:         ENABLED
loadsave:               ENABLED
```

```
screensizing:          ENABLED
lockout_optionring:    OFF
dozy_cheat:            ENABLED

//------------------------------------------------ Level Names.

Level Names
0:     0               Lara's Home
1:     12              Der kalte, kalte Krieg
2:     35              Level 2
3:     43              Level 3
4:     51              Level 4
5:     59              Level 5
Strings = 6
Buffer size = 67

//------------------------------------------------ Picture File Names.

Picture FileNames
Strings = 0
Buffer size = 0

//------------------------------------------------ Title File Names.

Title FileNames
0:     0               data\title.TR2
1:     15              data\title.pcx
2:     30              data\legal.pcx
3:     45              data\titleUS.pcx
4:     62              data\legalUS.pcx
5:     79              data\titleJAP.pcx
6:     97              data\legalJAP.pcx
Strings = 7
Buffer size = 115

//------------------------------------------------ Full Motion Video File Names.

FMV FileNames
0:     0               FMV\LOGO.RPL
Strings = 1
Buffer size = 13

//------------------------------------------------ Level File Names.

Level FileNames
0:     0               data\assault.tr2
1:     17              data\level1.TR2
2:     33              data\level2.TR2
3:     49              data\level3.TR2
4:     65              data\level4.TR2
5:     81              data\level5.TR2
Strings = 6
Buffer size = 97

//------------------------------------------------ Cutscene File Names.

Cutscene FileNames
Strings = 0
Buffer size = 0
```

```
//-------------------------------------------------- Script Sequences.

FRONTEND SEQ (SIZE: 6 bytes)
          FMV: FMV\LOGO.RPL
          ENDSEQUENCE

LEVEL0 SEQ (SIZE: 2 bytes)
          ENDSEQUENCE

LEVEL1 SEQ (SIZE: 12 bytes)
          SETTRACK: 33
          STARTLEVEL: Der kalte, kalte Krieg 'data\level1.TR2'
          LEVCOMPLETE
          ENDSEQUENCE

LEVEL2 SEQ (SIZE: 12 bytes)
          SETTRACK: 33
          STARTLEVEL: Level 2 'data\level2.TR2'
          LEVCOMPLETE
          ENDSEQUENCE

LEVEL3 SEQ (SIZE: 12 bytes)
          SETTRACK: 33
          STARTLEVEL: Level 3 'data\level3.TR2'
          LEVCOMPLETE
          ENDSEQUENCE

LEVEL4 SEQ (SIZE: 12 bytes)
          SETTRACK: 33
          STARTLEVEL: Level 4 'data\level4.TR2'
          LEVCOMPLETE
          ENDSEQUENCE

LEVEL5 SEQ (SIZE: 12 bytes)
          SETTRACK: 33
          STARTLEVEL: Level 5 'data\level5.TR2'
          LEVCOMPLETE
          ENDSEQUENCE

Total Sequence Data Size= 68 bytes
Writing FrontEnd Sequence

//-------------------------------------------------- Compiling.

Writing Level Sequences
Writing Demo Level Numbers
Demo Levels: None

//-------------------------------------------------- Game Strings (German).

Game Strings
0:        0                   Inventar
1:        9                   Option
2:        16                  Gegenst~ande
3:        29                  Game Over
4:        39                  Spiel laden
5:        51                  Spiel speichern
6:        67                  Neues Spiel
7:        79                  Abschnitt neu beginnen
8:        102                 Zur~uck zum Hauptmen~u
```

9:	125	Demo beenden
10:	138	Spiel beenden
11:	152	Abschnitt w~ahlen
12:	170	Position speichern
13:	189	Detailstufe w~ahlen
14:	209	Hoch
15:	214	Mittel
16:	221	Niedrig
17:	229	Gehen
18:	235	Rolle
19:	241	Laufen
20:	248	Links
21:	254	Rechts
22:	261	Zur~uck
23:	269	Schritt
24:	277	nach links
25:	288	Schritt
26:	296	nach rechts
27:	308	Umsehen
28:	316	Springen
29:	325	Handlung
30:	334	Waffe ziehen
31:	347	?
32:	349	Inventar
33:	358	Fackel
34:	365	Schritt
35:	373	Statistiken
36:	385	Pistolen
37:	394	Schrotflinte
38:	407	Automatik
39:	417	Uzis
40:	422	Harpune
41:	430	M16
42:	434	Granatwerfer
43:	447	Fackel
44:	454	Pistolen-Munition
45:	472	Schrot-Munition
46:	488	Automatik-Munition
47:	507	Uzi-Munition
48:	520	Pfeile
49:	527	M16-Munition
50:	540	Granaten
51:	549	Kleines Medi-Pack
52:	567	Gro=es Medi-Pack
53:	584	Aufnehmen
54:	594	Puzzle
55:	601	Schl~ussel
56:	612	Spiel
57:	618	Laras Haus
58:	629	LADE
59:	634	Ben~otigte Zeit
60:	650	Gefundene Geheimnisse
61:	672	Ort
62:	676	Besiegte Gegner
63:	692	Ben~otigte Munition
64:	712	Treffer
65:	720	Ben~otigte Spielst~ande
66:	744	Zur~uckgelegte Distanz
67:	767	Ben~otigte Medi-Packs
68:	789	Release Version 1.0

69:	809	Keine
70:	815	Beenden
71:	823	BESTZEITEN
72:	834	Keine Bestzeit
73:	849	Keine Daten
74:	861	Letzter Spielstand
75:	880	Spielstatistik
76:	895	von
77:	899	Was bisher geschah...
78:	921	spare
79:	927	spare
80:	933	spare
81:	939	spare
82:	945	spare
83:	951	spare
84:	957	spare
85:	963	spare
86:	969	spare
87:	975	spare
88:	981	spare

Strings = 89
Buffer size = 987

//--- PC Strings (German).

PC Strings

0:	0	Detailstufen
1:	13	Demo Modus
2:	24	Sound
3:	30	Steuerung
4:	40	Helligkeit
5:	51	Lautst~arke
6:	63	Eigene Belegung
7:	79	Die Daten konnten nicht gespeichert werden!
8:	123	Wiederholen?
9:	136	JA
10:	139	NEIN
11:	144	Gespeichert!
12:	157	Kein Spielstand vorhanden!
13:	184	Keine g~ultigen Daten
14:	206	Spiel speichern?
15:	223	- LEERER SLOT -
16:	239	AUS
17:	243	AN
18:	246	Soundkarten-Setup
19:	264	Standardeinstellung
20:	284	DOZY
21:	289	spare
22:	295	spare
23:	301	spare
24:	307	spare
25:	313	spare
26:	319	spare
27:	325	spare
28:	331	spare
29:	337	spare
30:	343	spare
31:	349	spare
32:	355	spare
33:	361	spare

```
34:     367             spare
35:     373             spare
36:     379             spare
37:     385             spare
38:     391             spare
39:     397             spare
40:     403             spare
Strings = 41
Buffer size = 409
```

//-- Puzzle, Pickup and Key Strings (German).

```
Puzzle1 Strings
0:      0               P1
1:      3               P1
2:      6               P1
3:      9               P1
4:      12              P1
5:      15              P1
Strings = 6
Buffer size = 18

Puzzle2 Strings
0:      0               P2
1:      3               P2
2:      6               P2
3:      9               P2
4:      12              P2
5:      15              P2
Strings = 6
Buffer size = 18

Puzzle3 Strings
0:      0               P3
1:      3               P3
2:      6               P3
3:      9               P3
4:      12              P3
5:      15              P3
Strings = 6
Buffer size = 18

Puzzle4 Strings
0:      0               P4
1:      3               P4
2:      6               P4
3:      9               P4
4:      12              P4
5:      15              P4
Strings = 6
Buffer size = 18

Pickup1 Strings
0:      0               P1
1:      3               P1
2:      6               P1
3:      9               P1
4:      12              P1
5:      15              P1
Strings = 6
```

```
Buffer size = 18

Pickup2 Strings
0:      0           P2
1:      3           P2
2:      6           P2
3:      9           P2
4:      12          P2
5:      15          P2
Strings = 6
Buffer size = 18

Key1 Strings
0:      0           K1
1:      3           Wachraum Schlussel
2:      22          GuardRoom Key
3:      36          GuardRoom Key
4:      50          GuardRoom Key
5:      64          GuardRoom Key
Strings = 6
Buffer size = 78

Key2 Strings
0:      0           K2
1:      3           Shift B-Taste
2:      17          Shaft B Key
3:      29          Shaft B Key
4:      41          Shaft B Key
5:      53          Shaft B Key
Strings = 6
Buffer size = 65

Key3 Strings
0:      0           K3
1:      3           K3
2:      6           K3
3:      9           K3
4:      12          K3
5:      15          K3
Strings = 6
Buffer size = 18

Key4 Strings
0:      0           K4
1:      3           K4
2:      6           K4
3:      9           K4
4:      12          K4
5:      15          K4
Strings = 6
Buffer size = 18
```

//-- end of the LOG file.

TR2 – THE STRINGS LIST

The input of GAMEFLOW consists of two files, one with the script commands and one with a list of strings which, in the published German version of Tomb Raider II Golden Mask demo, looks like this (-- comment separators added for clarity) (TABs readjusted for clarity):

//-- start of the STRINGS file.

// PC & PSX VERSION OF GENERAL GAME STRINGS

//-- Game Strings, 89 of them, padded with “spare”.

GAME_STRINGS:

```
                    Inventar
                    Option
                    Gegenst~ande
                    Game Over

                    Spiel laden
                    Spiel speichern
                    Neues Spiel
                    Abschnitt neu beginnen
                    Zur~uck zum Hauptmen~u
                    Demo beenden
                    Spiel beenden

                    Abschnitt w~ahlen
                    Position speichern

                    Detailstufe w~ahlen
                    Hoch
                    Mittel
                    Niedrig

                    Gehen
                    Rolle
                    Laufen
                    Links
                    Rechts
                    Zur~uck
                    Schritt
                    nach links
                    Schritt
                    nach rechts
                    Umsehen
                    Springen
                    Handlung
                    Waffe ziehen
                    ?
                    Inventar
                    Fackel
                    Schritt
```

```
                Statistiken
                Pistolen
                Schrotflinte
                Automatik
                Uzis
                Harpune
                M16
                Granatwerfer
                Fackel
                Pistolen-Munition
                Schrot-Munition
                Automatik-Munition
                Uzi-Munition
                Pfeile
                M16-Munition
                Granaten
                Kleines Medi-Pack
                Gro=es Medi-Pack
                Aufnehmen
                Puzzle
                Schl~ussel
                Spiel

                Laras Haus
                LADE

                Ben~otigte Zeit
                Gefundene Geheimnisse
                Ort
                Besiegte Gegner
                Ben~otigte Munition
                Treffer
                Ben~otigte Spielst~ande
                Zur~uckgelegte Distanz
                Ben~otigte Medi-Packs

                Release Version 1.0

                Keine
                Beenden
                BESTZEITEN
                Keine Bestzeit
                Keine Daten
                Letzter Spielstand
                Spielstatistik
                von
                Was bisher geschah...
        spare
        spare
        spare
        spare
        spare
        spare
        spare
        spare
        spare
        spare
        spare
END:
```

```
//------------------------------------------------- PC Strings, 41 of them, padded with "spare".

PC_STRINGS:

                Detailstufen
                Demo Modus
                Sound
                Steuerung
                Helligkeit
                Lautst~arke
                Eigene Belegung

                Die Daten konnten nicht gespeichert werden!
                Wiederholen?
                JA
                NEIN
                Gespeichert!
                Kein Spielstand vorhanden!
                Keine g~ultigen Daten
                Spiel speichern?
                - LEERER SLOT -

                AUS
                AN
                Soundkarten-Setup
                Standardeinstellung
                DOZY

        spare
        spare
        spare
        spare
        spare
        spare
        spare
        spare
        spare
        spare
        spare
        spare
        spare
        spare
        spare
        spare
        spare
        spare
        spare
        spare
END:
```

```
//---------------------------------------------------- PSX Strings, 80 of them, padded with “spare”.
//----------------------------------------------------             If omitted, the GAMEFLOW compiler will stop
//                                                                 with an error message:
//                                                                 ERROR: Incorrect number of psx strings.
//                                                                 0 strings, there should be 80
//---------------------------------------------------- However, they will not be present in the compiled file.

PSX_STRINGS:

                    Bildposition
                    DEMO
                    Sound
                    Steuerung
                    Gamma-Korrektur
                    Lautst~arke Einstellen
                    Steuerungsart
                    Fehler beim Sichern!
                    Nochmal?
                    JA
                    NEIN
                    Spielstand gesichert!
                    Kein Spielstand gefunden!
                    Kein Spielstand g~ultig!
                    Spiel sichern?
                    - LEERER SLOT -
                    Irgendeine Taste dr~ucken
                    Die Richtungstasten benutzen
                    um das Bild einzustellen
                    > W~ahlen
                    ; Zur~uck
                    > Weiter
                    Pause
                    Kein Controller
                    Spiel sichern auf
                    Spiel ~uberschreiben auf
                    Spielstand laden von
                    der Memory Card in
                    Slot 1
                    Sind Sie sicher?
                    JA
                    NEIN
                    ist voll
                    Es sind keine
                    Spielst~ande auf
                    Es ist ein Fehler auf
                    Es ist keine Memory Card in
                    ist nicht formatiert
                    M~ochten Sie sie
                    jetzt formatieren?
                    Sichere Spiel auf
                    Lade Spielstand von
                    Formatiere
                    Spiel wird ~uberschrieben
                    Spiel sichern
                    Spiel laden
                    Memory Card formatieren
                    Verlassen
                    Weiter
                    Sie k~onnen keine Spiele
                    sichern ohne eine
                    formatierte Memory Card
                    mit mindestens einem
                    freien Block einzulegen
                    Die Memory Card in
                    Verlassen ohne zu sichern
                    Verlassen ohne zu laden
                    Spiel starten
                    UNFORMATIERTE MEMORY CARD
                    Legen Sie eine formatierte Memory Card ein
                    oder dr~ucken Sie > um ohne zu
                    Sichern fortzufahren.

// PAL CODES: ENGLISH= SLES-00718, FRENCH= SLES-0719, German= SLES-0720

                    bu00::BESLES-00720TOMB2
// NTSC CODES: US= UNKNOWN, JAPAN= UNKNOWN

                    bu00::BASLUS-00152TOMB2

          spare
          spare
          spare
          spare
          spare
          spare
          spare
          spare
          spare
          spare
          spare
          spare
          spare
          spare
          spare
          spare
END:

//-------------------------------------------------- end of the STRINGS file.
```

TR2 – THE SCRIPT TEXT

The input of GAMEFLOW consists of two files, one with the script commands and one with a list of strings. The file with the script commands is the one that matters the most for the researcher. That's where the names of the levels, the filenames, the game flow, etc, are determined. That's where the puzzles, pickups and keys are declared. The published German version of Tomb Raider II Golden Mask demo, looks like this (//-- comment separators added for clarity)(TABs readjusted for clarity)(re-arranged to match the compiled DAT file) (// commented where appropriated):

```
//------------------------------------------------ start of the SCRIPT TEXT file.

Description:    Tomb Raider II Script. Final Release Version 1.1 (c) Core Design Ltd 1997

//------------------------------------------------ Options.

// Option Types
//          EXIT_TO_TITLE
//          LEVEL%d
//          DEMO%d
//          SEQUENCE%d
//          EXITGAME
//------------------------------------------------------------------------
//                    O P T I O N S
//------------------------------------------------------------------------

Options:                                            // Defaults

//firstOption:                  LEVEL 1             // EXIT_TO_TITLE

//title_replace:                LEVEL 1             // -1

//ondeath_demo_mode:            LEVEL 1             // EXIT_TO_TITLE

//ondeath_ingame:               LEVEL 1             // EXIT_TO_TITLE

//noinput_time:                 150                 // 900

//on_demo_interrupt:            LEVEL 1             // EXIT_TO_TITLE

//on_demo_end:                  DEMO1               // EXIT_TO_TITLE

singlelevel:                    1                   // -1

//enable_cheat_code:                                // Not shown in the original script.

//select_any_level:

//use_security_tag:                                 // Not recognized as a valid command !!!
                                                    // Set by the compiler when a cypher is used.

dozy_cheat_enabled:                                 // What is this, in a TR2 level?

//lockout_optionring:
```

```
//screensizing_disabled:

//loadsave_disabled:

//noinput_timeout:

//cheatmodecheck_disabled:

//title_disabled:

//demoversion:

cypher_code:                166          // using a cypher sets  the use_security_tag.

Language:                   ENGLISH      // ENGLISH

Secret_Track:               47

//enable_indoor_reverb:                  // NO // Not recognized as a valid command !!!

//enable_cheat_key:                      // Not recognized as a valid command !!!

end:


//--------------------------------------------------- Title File Names: the TR2 level and the PCX pictures.

//------------------------------------------------------------------------
//                    T I T L E   S E T U P
//------------------------------------------------------------------------

Title:

Game:           data\title.tr2           // First file in 'Title' must always be title.tr2

PSXfile:pixUK\title.raw                  // PSX file names declared here as well.
PSXfile:pixUK\legal.raw                  //
PSXfile:pixUS\titleUS.raw                //
PSXfile:pixUS\legalUS.raw                //
PSXfile:pixJAP\titleJAP.raw              //
PSXfile:pixJAP\legalJAP.raw              // ... but they can be omitted.

PCfile:         data\title.pcx
PCfile:         data\legal.pcx
PCfile:         data\titleUS.pcx         // Foreign file names declared here as well.
PCfile:         data\legalUS.pcx         //
PCfile:         data\titleJAP.pcx        //
PCfile:         data\legalJAP.pcx        // ... but they can be omitted.

Track:          64                       // Title Sound Track ID.

end:
```

```
//------------------------------------------------ The visible part of the script: the Frontend and the Levels.

//-----------------------------------------------------------------------
//          F R O N T E N D
//-----------------------------------------------------------------------

Frontend:

fmv_start:          1                                  // Starting frame of the video. NOT USED.
fmv_end:            862                                // Ending frame of the video. NOT USED.
fmv:                fmv\logo.rpl                       // File name of the FMV.

end:


//-----------------------------------------------------------------------
//          ASSAULT        L A R A ' S   H O M E
//-----------------------------------------------------------------------

gym:                Lara's Home                        // This is not called a "level", it is a "gym".

//Secrets:          0
//Load_Pic:         pix\mansion.raw
//track:            0
//game:             data\assault.tr2

end:


//-----------------------------------------------------------------------
//          LEVELS
//-----------------------------------------------------------------------

Level:              Der kalte, kalte Krieg
track:              33                                 //Ambient (Wind)
Load_Pic:           pix\pic1.raw
game:               data\level1.tr2
key1:               Wachraum Schlussel
key2:               Shift B-Taste
complete:
end:

Level:              Level 2
track:              33                                 //Ambient (Wind)
Load_Pic:           pix\pic1.raw
game:               data\level2.tr2
key1:               GuardRoom Key
key2:               Shaft B Key
complete:
end:

Level:              Level 3
track:              33                                 //Ambient (Wind)
Load_Pic:           pix\pic1.raw
game:               data\level3.tr2
key1:               GuardRoom Key
key2:               Shaft B Key
complete:
end:
```

```
Level:          Level 4
track:          33                       //Ambient (Wind)
Load_Pic:       pix\pic1.raw
game:           data\level4.tr2
key1:           GuardRoom Key
key2:           Shaft B Key
complete:
end:

Level:          Level 5
track:          33                       //Ambient (Wind)
Load_Pic:       pix\pic1.raw
game:           data\level5.tr2
key1:           GuardRoom Key
key2:           Shaft B Key
complete:
end:

//------------------------------------------------ The game and pc strings are in a separate file.

GameStrings:    strings.txt

//------------------------------------------------ end of the SCRIPT TEXT file.
```